Integration and deployment of TwinEU platform

Deliverable No: D4.1
Work package: WP4

Official delivery date: 30.06.2025
Actual delivery date: 04.07.2025

Dissemination level: Public

Co-funded by the European Union
Project: 101136119 | HORIZON-CL5-2023-D3-01 | www.twineu.net

Disclaimer

Funded by the European Union. 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.

Document Authors	
Lenos Peratitis (ED)	Vasilis Theodorou (UBE)
Apostolos Kapetanios (ED)	Ferdinando Bosco (ENG)
Dimitrios Apostolidis (ED)	Luca Greci (ENG)

Version	Date	Author(s)	Notes
0.1	16/08/2025	Lenos Peratitis (ED)	Table of Contents
0.7	03/06/2025	ED	First Draft of Document
0.8	18/06/2025	All D4.1 Contributors	Finalized Contributions
0.9	26/06/2025	ELES, RWTH	Reviewed
1.0	03/07/2025	ED	Final Draft for Submission

	Name Surname	Date
Responsible Partner	European Dynamics	
Checked by WP leader	Lenos Peratitis (ED)	03/07/2025
Verified by the appointed Reviewers	Gregor Omahen (ELES) Juan Adolfo Galeano (RWTH)	24/06/2025 26/06/2025
Approved by Project Coordinator	Padraic McKeever (Fraunhofer)	04/07/2025

Table of Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

Acronym	Meaning
AI	Artificial Intelligence
AIOTI	Alliance for IoT and Edge Computing Innovation
BDVA	Big Data Value Association
BD4NRG	Big Data for Next Generation Energy
COSMAG	Comprehensive Architecture for Smart Grid
CPU	Central Processing Unit
DSO	Distribution System Operator
DSP	Dataspace Protocol
DT	Digital Twin
EU	European Union
GA	Grant Agreement
GUI	Graphical User Interface
HLA	High Level Architecture
IAM	Identity and Access Management
IDSA	International Data Space Association
ISO	International Organization for Standardization
ML	Machine Learning
MS	Milestone
MVP	Minimum Viable Product
NGO	Netcode for GameObjects
OS	Operating System
PUN	Photon Unity Networking
RA	Reference Architecture
RAM	Reference Architecture Model
SDK	Software Development Kit
UI	User Interface
TSO	Transmission System Operator
VR	Virtual Reality
WP	Work Package
XR	Extended Reality
XRCU	XR Client for Unity

Executive Summary

The TwinEU platform development focuses on the implementation and integration of a Middleware that enables the secure, interoperable and scalable federation of Digital Twin systems across Europe's energy ecosystem. This development process is based on the combined analysis of the defined Data-Space-enabled Middleware architecture and requirements alongside with the analysis of TwinEU Digital Twin use cases.

The decentralized TwinEU platform (Middleware) is designed and developed in order to integrate the value-added modules of the TwinEU whole framework namely, Services Workbench/ Big Data / AI marketplace, Interactive augmented exploration of TwinEU DT and the TwinEU Digital Twins pan-European network.

The TwinEU platform's main objective is to operate as the core, administrative component of the broader TwinEU Framework. It implements functionalities such as robust authentication of TwinEU participants, thereby ensuring adherence to established data access control policies, discovery functionalities and privacy regulations. Additionally, it serves as the primary access interface to the broader TwinEU Framework, facilitating the configuration and integration of secure, peer-to-peer communication channels among participants (through the TwinEU Connector). Finally, the TwinEU Middleware contains the administrative and executable components that are fully compatible with the OneNet Connector (the base of the TwinEU Connector) and the European Energy Data Space that manage and coordinate the interoperability, sovereignty, robustness and efficiency within the IDSA network, alongside with the support of seamless interaction with external actors and services.

The first development process implemented until TwinEU M18 presents the following outcomes:

- A robust middleware infrastructure that enables interoperability and integration among all the TwinEU Framework modules.
- A scalable orchestration mechanism for real-time installation, configuration and interaction of distributed Digital Twins through the TwinEU Connector.
- Integration guidelines to align various tools, data models, and computational environments into a unified deployment strategy.
- A clear roadmap for the platform's evolution, with flexibility to incorporate future TwinEU services, models, and system extensions.
- Integration with the Data Space federated network in order to manage the lifecycle of digital assets, ensuring secure and sovereign data exchange.

In general, the TwinEU platform develops a scalable, modular, and interoperable Middleware that supports real-time data and model exchange, multi-level orchestration, and secures interaction between heterogeneous Digital Twin instances, thereby facilitating the deployment of pan-European energy scenarios.

The results of the first version of TwinEU platform (M18) demonstrate that federated Digital Twin architecture is both technically feasible and promising operationally. By enabling a distributed but coordinated approach to Digital Twin coordination and integration, the TwinEU platform lays the foundation for enhanced observability, predictive analytics, and collaborative decision-making across Europe's energy systems, supporting the strategic objectives of resilience, sustainability, and cross-border energy cooperation.

1 Introduction

Europe is undergoing a transformative shift in its energy landscape, driven by the urgent need to enhance sustainability, security, and resilience in the face of geopolitical, environmental, and economic pressures. The increasing integration of renewable energy sources, decentralised energy production, and evolving market dynamics demand a more intelligent, interoperable, and coordinated infrastructure. In this context, Digital Twins (DTs) emerge as powerful tools for enabling real-time visibility, predictive insights, and intelligent coordination across energy networks in real time. The concept of federated Digital Twins, which is introduced by the TwinEU project, addresses the need for scalable, decentralised digital infrastructures that respect data sovereignty, support cross-border energy coordination, and enable advanced simulation, forecasting, and decision-making capabilities through AI and high-performance computing.

The TwinEU Platform Development lies at the heart of this transformation. It focuses on building the core platform that connects diverse local and national DT instances into a unified, TwinEU pan-European framework. This development process includes the development of the TwinEU middleware, integration of data and model-sharing mechanisms, and deployment of the Data-Space-enabled federated network of Digital Twin. The TwinEU platform acts as the orchestrator of distributed resources, services, and scenarios across this federated architecture. Additionally, the TwinEU Middleware operates as an administrator of the value-adding functionalities of the TwinEU ecosystem such as the Services Workbench/ Big Data / AI marketplace for interoperability and Interactive augmented exploration tools of TwinEU DT.

This deliverable presents the work that has been done regarding the activities of Task 4.4 (TwinEU Platform Development) and the correlated work done in Tasks 4.1, 4.2 and 4.5. It provides thorough guidelines and methodologies regarding the implementation and integration of scalable, decentralized digital infrastructures that enable federated data and model exchange, respect data sovereignty, support cross-border energy coordination, and enable advanced simulation, forecasting, and decision-making capabilities through AI and high-performance computing.

The purpose of Task 4.4 is to design, implement, and validate a modular, scalable, and secure platform capable of hosting and coordinating multiple Data Space participants, services and Digital Twins. It aims to enable cross-domain and cross-border collaboration while ensuring data governance, computational efficiency, and seamless interaction with external services and actors.

To explore how this initiative is being realised, this document will present:

- The underlying position of the Middleware in the TwinEU reference architecture of the federated Digital Twin framework.
- The components of the TwinEU platform, along with their functionality, development process and integration.
- The integration of core services like the Service Workbench, AI/Big Data marketplace, and interactive DT exploration tools with the TwinEU Middleware.
- The adaptation of the European Data Space concept from the TwinEU Framework to support secure, interoperable model and data sharing.
- The implementation plan and integration methodology for supporting TwinEU Framework functionality and pan-European energy scenarios.

Through these efforts, Task 4.4 delivers the technological backbone of the TwinEU vision — a platform not only for today's needs but also for Europe’s digital energy future.

1.1 Task 4.1 Services Workbench/ Big Data / AI marketplace for interoperability

The primary objective of Task 4.1 is to establish the foundational Big Data management capabilities of the TwinEU platform, ensuring efficient batch and real-time data ingestion, management, and curation processes. A critical goal is enabling seamless integration of advanced AI technologies that address key data-driven challenges in the energy sector, including accurate demand and generation forecasting, behavioural analytics, data handling, visualization and elasticity profiling. The TwinEU middleware serves as the functional enabler for the effective and interoperable integration and facilitation of the Services Workbench and AI Marketplace, enabling the deployment of intelligent, modular, and scalable services across a federated European Digital Twin infrastructure.

1.2 Task 4.2 Interactive augmented exploration of TwinEU DT

The core objective of Task 4.2 is to develop a scalable, adaptable and user-friendly multiuser Extended Reality (XR) environment capable of delivering high-quality immersive experiences regarding the management and visualisation of TwinEU Digital Twin systems. Task 4.2 will also provide advanced data and model visualization services designed to enhance interaction and collaborative workflows for stakeholders within the TwinEU ecosystem. The XR environment that will be developed should be compatible and pluggable with the TwinEU platform. The TwinEU platform will offer administrative functionalities, offering, discovery, management, data access and role-authorization control to the TwinEU users that are willing to use and create assets that are part of this XR environment.

1.3 Task 4.4 TwinEU platform development

Task 4.4 delivers the TwinEU middleware platform - a scalable integration layer that unifies data, digital models, and compute resources (see Figure 1). It starts with deploying foundational frameworks and performing gap analyses to define MVP components, then folds existing tools and services into a coherent stack. Central to the effort is the Digital Twin Federator, a data-space connector that clusters local twins into a pan-European ecosystem.

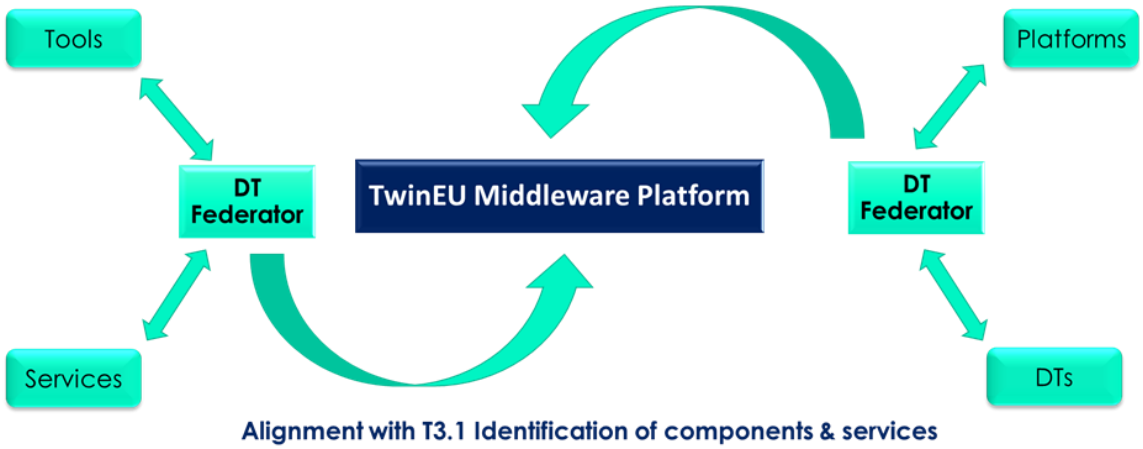

Figure 1: Middleware Integration Methodology

1.4 Task 4.5 EU Data Space adaptation to support the TwinEU continuum

The main objective of Task 4.5 is to ensure that the TwinEU platform, the broader TwinEU framework and the TwinEU participants are perfectly aligned with EU Data Space principles. It focusses on robust digital model sharing, comprehensive data governance, and secure, trusted, and interoperable data management. The practical scope of this task is to iteratively design, implement and enhance the TwinEU Data-Space Connector that is based on the defined and robust OneNet Connector. The TwinEU connector should be developed in order to serve the best way the TwinEU use cases. Finally, the TwinEU Connector has to be perfectly compatible with the TwinEU Platform that will administrate and offer management functionalities for the operation of the Data Space Connector federated network.

1.5 Objectives of the Work Reported in this Deliverable

The objective of the work reported in this deliverable is to thoroughly set the principles and implement the first iteration (M18) of the development and integration of the TwinEU platform. The work includes the definition of the relationship of the TwinEU Middleware with the value-added modules of the TwinEU whole framework namely, Services Workbench/ Big Data / AI marketplace, Interactive augmented exploration of TwinEU DT and the TwinEU Digital Twins pan-European network. The work defines the interfaces and the integration prerequisites, process and integration methodology between the components.

Analytically, the objectives of the activities that are described in this document are the following:

- Establishing a robust middleware layer that supports and administrates decentralized data and model sharing across heterogeneous Connectors and stakeholders.
- Ensuring compatibility with data space standards and OneNet connectors to enable secure and sovereign data/model exchange.
- Develop a decentralized Middleware responsible for orchestrating the dynamic management of local DTs, managing interoperability, and supporting scaling towards a fully operational pan-European Digital Twin ecosystem.
- Laying the technical foundation to enable pan-European use cases and cross-border coordination of energy infrastructure through shared Workbench/ Big Data / AI marketplaces and Interactive augmented exploration of Digital Twins.
- Create extensive documentation for potential TwinEU developers, users and generally participants to develop and integrate the desired software infrastructure and be prepared for the next development iterations.

1.6 Outline of the deliverable

The remainder of Deliverable 4.1 is structured as follows:

- **Chapter 2 – Reference Architecture & The TwinEU Platform** introduces the TwinEU Reference Architecture, providing a comprehensive description of its design principles, general structure, technological stack, and alignment with existing reference frameworks (connection with D3.1).

- **Chapter 3 – Services Workbench/Big Data/AI Marketplace for Interoperability** elaborates on the Services Workbench and Big Data/AI Marketplace by describing their main software

components, specific functionalities, and the planned approach for their deployment and operational integration.

- **Chapter 4 – Interactive Augmented Exploration of TwinEU DT** defines the approach for developing the interactive, augmented exploration framework for the TwinEU Digital Twin, emphasizing immersive XR interactions, detailed component descriptions, key features, and strategies for effective deployment.
- **Chapter 5 – Data Space Adaptation to Support the TwinEU Continuum** details how the TwinEU implementation aligns and integrates with EU Data Space standards, clearly outlining component-level descriptions, functional capabilities, and specific deployment approaches.
- **Chapter 6 – TwinEU Integrated Platform Development provides** comprehensive integration guidelines for the TwinEU middleware platform, specifying the features and functionalities of its core components, defining relevant testing scenarios, and presenting both the status and future development plans.
- **Chapter 7 – Development and Integration methodology and plan.**
- **Chapter 8 – Conclusions** summarize the key results in this deliverable and outlines next steps.

1.7 How to Read this Document

To effectively understand this deliverable, readers should first review Deliverable D3.1 57[18], which establishes the foundational TwinEU Reference Architecture and provides essential context on system functionalities, architectural decisions, and interoperability frameworks. Additionally, familiarity with Deliverable D2.2 “TwinEU use cases, pan-European scenarios and KPIs” [32], is recommended for deeper insight.

This document systematically builds upon these prior insights to present a coherent deployment plan for the TwinEU middleware platform, laying the groundwork for future technical integration and validation activities detailed in D4.3 "Validation of Platform & Implementation of TwinEU Pan-European Scenarios."

2 Reference Architecture & The TwinEU platform

This section provides a summarized description of the TwinEU Reference Architecture 57[18], and the individual components that constitute it. Additionally, there will be a brief description of the functionality and the core attributes of these components in parallel with the relationships between them.

2.1 Introduction

The TwinEU project aims to establish a federated pan-European DTs ecosystem, based on a robust, modular and resilient Reference Architecture. The core objective of this Reference Architecture is to enable the smooth integration of renewable energy sources and promote sustainable energy management across Europe. The TwinEU Reference Architecture is designed to support the economic, societal and technical objectives of the TwinEU project, comply with European policies and restrictions and address the challenges regarding the construction of an interoperable, scalable framework of multiple and heterogeneous stakeholders of the energy domain. Specifically, the TwinEU Reference Architecture acts as a standardized ecosystem that enables the integration of DTs and related services, supports the coexistence and collaboration of various energy actors and facilitates the data and model sharing among them. This federated ecosystem serves stakeholders such as DSOs, TSOs, prosumers, network operators, aggregators, market operators and other relevant actors of the energy domain and provides the capabilities of simulation, modelling, control and analysis of Europe's energy network.

The Reference Architecture aligns with important European initiatives such as GAIA-X [1], FIWARE [2], BRIDGE [3] and IDSA [4], integrates open standards such as NGSI-LD APIs and is based on past projects such as OneNet [5], BD4NRG [6] and ENERSHARE [7] to deliver an updated, compatible architectural framework. From an operational standpoint the TwinEU Architecture provides a middleware (based on the OneNet framework middleware) that coordinates the ecosystem of federated DTs integrated with IDSA Connectors that ensure interoperability between systems such as the Continental Europe Synchronous Area and local energy communities, facilitating coordinated operation between transmission and distribution networks.

The core functionality of the federated system enables advanced scenario modelling and simulation capabilities, performs detailed grid behaviour analysis and helps stakeholders assess the large-scale integration of renewable energy sources. The TwinEU RA supports the development of innovative services and business models by fostering an open, interoperable ecosystem. This environment encourages entrepreneurship and accelerates the deployment of new, data-driven solutions, strengthening the resilience of the energy economy. Its open-source nature promotes broad adoption and allows grid operators and market players to implement scalable and adaptable technologies. From a societal standpoint, the architecture boosts grid resilience through dynamic monitoring and improved system management. It integrates tools for forecasting, anomaly detection, and the identification of infrastructure constraints, enabling proactive planning and operation. These capabilities enhance system flexibility, reduce operational costs, and minimize environmental impact, thereby ensuring a reliable and sustainable energy supply. In essence, the TwinEU Reference Architecture serves as more than just a technical foundation—it is a transformative driver of economic growth, societal progress, and environmental sustainability. By aligning with European strategies and embracing state-of-the-art technologies, it firmly positions TwinEU at the forefront of Europe's energy transition.

2.2 TwinEU Reference Architecture

The methodological approach of the TwinEU Reference Architecture [18] is based on the 4+1 Architectural UML view model and the ISO 42010 [8] standard, and follows a sequence of steps regarding its implementation. To effectively describe software architecture, it is common to use models that capture multiple perspectives of the system. One widely adopted approach in UML is the "4 + 1" view model [9], which provides a comprehensive representation by organizing the architecture into five interconnected views. Each view contributes to a unique perspective, collectively offering a holistic understanding of the system. The five key views of the "4 + 1" model are:

- **Logical View:** Focuses on the system's functional requirements, especially from the end-user perspective. It defines key elements such as objects, components, and classes.
- **Process View:** Captures the dynamic aspects of the system, including its runtime behaviour, concurrency, and communication between processes. It addresses performance and scalability concerns.
- **Physical View:** Illustrates how the software components are deployed onto the hardware infrastructure, providing insight from a system engineering standpoint.
- **Development View**: Describes the software's static structure, showing how it is organized in terms of modules and components from the developer's perspective.
- **Scenarios (use cases):** Serve to integrate and validate the four previous views by demonstrating how different parts of the system collaborate to fulfil specific use cases.

ISO 42010:2022 is an international standard that offers a structured framework for developing and managing architecture descriptions of complex systems. It establishes the essential elements of system architecture and outlines how these elements address various stakeholder concerns. ISO 42010 identifies three core components that support effective architecture description:

- Architecture Description Frameworks define the structure and conventions used to describe architectures.
- Architecture Description Languages provide the formal language and syntax for expressing architectural concepts.
- Architecture Viewpoints and Views help capture and organize stakeholder concerns by structuring the design process into different perspectives.

Given its multi-layered operation within the broader energy system, the TwinEU ecosystem represents a complex system where the structured approach defined by ISO 42010 can be particularly beneficial in managing its architectural complexity. The TwinEU project follows specific steps in order to design and implement the final Reference Architecture. These steps, in order of priority include:

- Constructing an initial architecture draft for review by the responsible partners
- Conducting a bottom-up and top-down analysis based on the use cases, requirements, specifications and objectives,
- Integrating results relevant solutions from other European projects in the updated draft; and
- Delivering the final version of the Reference Architecture.

TwinEU Reference Architecture is influenced by architecture models such as "BRIDGE DERA Reference Architecture" Data Management Working Group [10], Smart Grid Architecture Model (SGAM) [11], Comprehensive Architecture for Smart Grid (COSMAG) [12], FIWARE Smart Energy Reference Architecture [13] and Alliance of Internet of Things Innovation (AIOTI) High Level Architecture [14] and makes use of multiple component frameworks for data processing across Europe

such as the International Data Spaces Association (IDSA), the Big Data Value Association (BDVA) [15] and the GAIA-X project [16].

2.3 TwinEU General Functionality

The core structure of the Reference Architecture consists of three pillars: the Middleware, the OneNet DT Connector (coupled with DT entities) and the DT Services Workbench and AI Marketplace. The TwinEU Middleware is the core component of a federated system that performs distributed data exchange. Specifically, the TwinEU Middleware ensures cohesion and collaboration among the distributed OneNet DT Connectors which are embedded and serve the data exchange between DTs. The Middleware is responsible for the following functionalities:

- Configures and Integrates new Connectors for the TwinEU stakeholders.
- Supports federated authentication and authorization, allowing participants to retain control over their data and identities while enabling secure, role-based access across services.
- Supports interoperability defining the principles and standards that should be followed by the TwinEU stakeholders.
- Facilitates the orchestration of distributed services and allows users to discover and consume services offered by other members of the TwinEU.
- Provides a collection of metrics, logs, and traces from all participating systems for performance monitoring and compliance auditing.

The TwinEU Connectors serve as the trusted technical enablers for secure, sovereign, and controlled data exchange between independent stakeholders. They form the backbone of data interoperability and governance within the decentralized system. The Connector is responsible for the following functionalities:

- Ensures the principle of data sovereignty, namely the right of data providers to retain full control over how, when, and by whom their data is accessed.
- Enables standardized communication between heterogeneous systems. By following a common protocol stack and semantic model, they allow the stakeholders in a federated network to exchange data regardless of differences in software they use or data formats.
- Ensures trust and security, performing operations such as authentication and authorization, secure communication and identity validation.
- Enforces data usage policies defined by the data provider.
- Enables peer-to-peer, decentralized data exchange. Each TwinEU stakeholder can potentially run its own Connector, forming a federation of autonomous nodes.
- Offer extensibility and scalability, as it is designed to integrate with existing enterprise systems and platforms.

The Services Workbench and AI Marketplace are a framework that comes to support and offer added value to the TwinEU ecosystem. As specific components will offer a wide range of opportunities for the TwinEU stakeholders to fully leverage their data, their models and their assets, receiving crucial outcomes and advantages thus maximizing and improving their processes.

2.4 The TwinEU Architecture

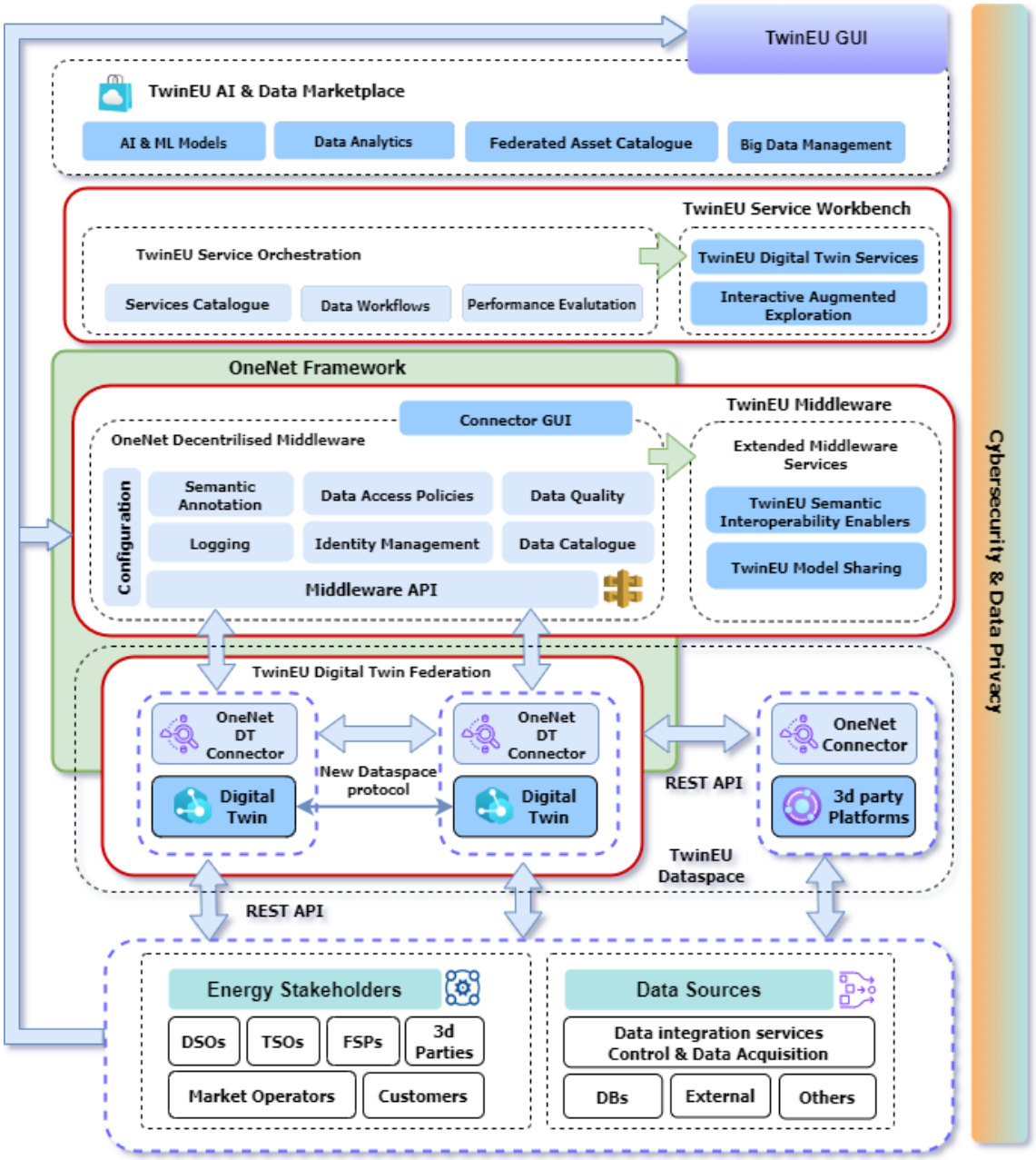

Figure 2: TwinEU RA [18]

The final TwinEU Reference Architecture design is based on the below four pillars:

1. **DT Federation:** RA enables **secure, decentralized collaboration among energy stakeholders** (TSOs, DSOs, Market Operators, Aggregators) by orchestrating interoperability between their DT systems.
2. **Data Space integration:** RA implements the core principles of European Data Spaces, ensuring **secure, sovereign, and policy-driven** data exchange between energy stakeholders.
3. **Model Sharing and Reusability:** TwinEU RA facilitates the reuse and exchange of **DT models** across domains and applications through standardized interfaces.

4. **Semantic Interoperability:** RA ensures semantic alignment of exchanged data by adopting **common vocabularies, ontologies, and data models.**

Figure 2 highlights the simplified Reference Architecture where the specific WP4 TwinEU platform components can be mapped. This design also validates the TwinEU platform compliance with the envisioned architecture.

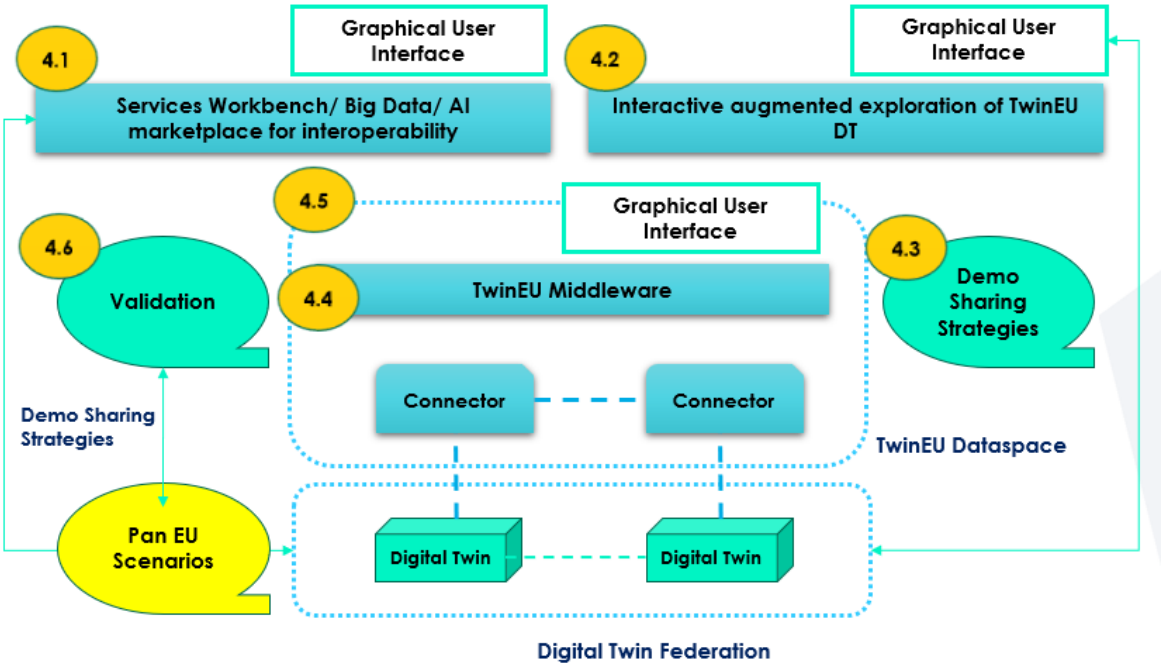

Figure 3: TwinEU platform components aligned to Reference Architecture

3 Services Workbench/ Big Data / AI marketplace for interoperability

Task 4.1 introduces a set of interoperable and modular components that collectively support the data-centric and service-oriented needs of the TwinEU federated ecosystem, maximizing the utility and accessibility for stakeholders. The objective is to facilitate seamless ingestion, curation, discovery, and usage of data and services across the distributed DT environment.

3.1 Components Description

To address the evolving challenges of energy data processing, AI service integration, and cross-platform interoperability, four core components have been identified and described below. Each component plays a distinct role, but they operate in constructive collaboration, forming foundational elements of the TwinEU architecture. This task introduces four core components, namely:

- Big Data Management Module
- AI Models Integration component
- Interoperable Marketplace
- Services Workbench

The Big Data Management Module is one of the core components, which is responsible for supporting both batch and real-time data ingestion workflows, data storage, and curation processes. Leveraging MinIO's [17] object storage capabilities, the module efficiently manages unstructured data and complex digital assets, including raw datasets, logs, and serialized AI models. MinIO's support for rich metadata and tagging enhances data discoverability and governance, facilitating seamless integration with AI workflows and downstream services. Its scalable architecture ensures high performance and reliability, meeting the demands of diverse energy sector use cases. Additional technologies considered for this module include RabbitMQ [19], used for event-driven data ingestion, and Apache NiFi [20], a flow-based tool for orchestrating complex data pipelines. The module supports clean, reliable, and policy-compliant data circulation across the TwinEU platform.

Building upon the foundational role of the Big Data Management Module, the AI Models Integration component facilitates the discoverability and reuse of AI models developed to address data-driven challenges, such as forecasting, elasticity profiling, and behavioural analytics. The emphasis of this component is on registering and exposing these models through structured metadata, enabling stakeholders to locate relevant tools that match their analytical needs and available datasets. Its role is to provide a federated view into AI capabilities across the TwinEU ecosystem, ensuring traceability, interoperability, and integration readiness.

To enable a broader interaction with the available data and services, the Interoperable Marketplace acts as the interface for discovery and access. This marketplace extends the concept of a traditional federated catalog by enriching it with metadata about services, datasets, and AI models enhancing searchability and access control. Aligned with IDSA principles, it allows stakeholders to explore interoperable offerings through a unified interface, supporting an elevated level of automation and governance over access and usage rights.

Finally, the Services Workbench serves as the orchestration and execution layer where containerized services can be deployed, managed, and monitored. Building upon the open-source implementation delivered in the OneNet project, the Workbench includes all necessary components

to enable the integration of data and services within a common execution environment. It provides a user interface for service deployment and configuration, incorporates Kubernetes for orchestration, and supports the secure exchange of data through integration with the federated connectors. This component plays a crucial role in realizing a plug-and-play architecture where analytics and visualization services can be easily instantiated and reused across use cases and stakeholder environments.

3.2 Components Functionalities

The functionalities of the components form a comprehensive data and service infrastructure that supports the core operational needs of the TwinEU platform. Together, they enable structured data ingestion, enriched metadata discoverability, secure service orchestration, and continuous monitoring and evaluation, while remaining extensible and interoperable across the federated ecosystem.

- **Data Ingestion and Management:** The Big Data Management Module accommodates both real-time and batch data ingestion. Real-time ingestion is supported through message queues using RabbitMQ, while batch data upload is enabled via direct interaction with MinIO's S3-compatible API. Apache NiFi is optionally used to orchestrate complex flow-based data pipelines across multiple sources and formats. Additionally, metadata tagging and lifecycle management are applied to incoming datasets to ensure traceability, management and compliance with usage policies.
- **Model and Service Discoverability:** Discoverability is enabled through a federated catalogue that spans AI models, datasets, and services. Metadata schemas are used to describe technical and functional characteristics such as capabilities, input/output formats, and access conditions. Descriptive metadata and field-level annotations allow precise filtering and effective model/service matchmaking based on stakeholder needs. The marketplace interface supports attribute-based access control, aligning with the interoperability and governance objectives promoted by IDSA.
- **Containerized Execution and Evaluation:** The Services Workbench supports dynamic deployment and execution of containerized services. Services are orchestrated using Kubernetes and Helm, building on the OneNet Workbench foundation. A user interface allows stakeholders to configure services, select relevant datasets, and define execution parameters. During runtime, the platform enables performance monitoring and resource usage tracking, supporting service evaluation, reproducibility, and auditability.
- **Visualization and Analytics:** Integrated observability tools allow platform users to analyse outcomes and monitor system behaviour in real time. Grafana and Prometheus dashboards provide visual access to service outputs, system health indicators, and operational trends.
- Extensibility and Security: The TwinEU architecture is designed for modular expansion and secure integration. Services can be onboarded with minimal configuration, ensuring continued interoperability across the platform. Secure APIs enforce authentication, authorization, and data usage policies. The Big Data Management component, based on MinIO, is protected using Keycloak as its Identity and Access Management (IAM) module, enabling fine-grained access control. The overall design supports trusted and sovereign data sharing among participants.

3.3 Deployment

The deployment strategy adopted in this task follows a modular, container-based architecture that supports flexibility, scalability, and interoperability across the TwinEU ecosystem. It is designed to enable the seamless integration of the core components—Big Data Management Module, AI model

integration, Interoperable Marketplace, and Services Workbench—within heterogeneous environments and infrastructures.

A dockerized approach is being followed to encapsulate each component into self-contained services, ensuring consistency in configuration, portability, and reproducibility. The architecture facilitates progressive deployment, allowing each component to be integrated and evolve independently while remaining aligned with the overarching interoperability framework of the platform.

The deployment setup supports infrastructure-agnostic hosting, enabling partners to instantiate components locally or in cloud environments, depending on their operational requirements. Configuration parameters are externalized, promoting ease of adaptation and future extension. As development progresses, additional containers, orchestration scripts, and configuration mechanisms will be introduced to realize the full vision of a unified, interoperable, and secure data and service layer. This approach ensures that the TwinEU platform remains extensible and future-ready, supporting continuous integration of new services, evolving stakeholder needs, and the federated operation of DTs at both local and pan-European levels.

4 Interactive augmented exploration of TwinEU DT

Virtual Reality (VR) has emerged as a transformative medium for exploring complex data and environments through immersive, interactive experiences. Real-time VR visualization applications require high-performance systems to maintain a sense of presence and fluid interaction. This chapter describes the implementation of the XR Framework of TwinEU Platform, which supports high-quality, real-time, immersive DT visualization experiences. The main objectives of the XR Framework are:

- Enable multiuser XR environments for enhanced collaboration and decision-making.
- Facilitate interactive data visualization using VR approach.
- Enhance data accessibility and usability through advanced visualization and validation services.
- Integrate Unity3D-based immersive workbench as a plugin, providing an intuitive and simplified workflow.

4.1 Components Description

The preliminary activity for the design of the XR Framework was an in-depth analysis about XR technologies for the implementation of a collaborative multi-user visualization and validation framework. For the scope of the framework the VR technology has been taken into consideration.

4.1.1 VR Technologies: State of the Art and main requirements

4.1.1.1 Key Features of Real-Time VR Visualization Applications

Real-time VR visualization systems are defined by the following essential capabilities:

- **Immersive 3D Environment**
  Users are placed within stereoscopic, spatial environments that deliver a strong sense of depth and scale, surpassing the limitations of traditional 2D interfaces.
- **Real-Time Rendering**
  High frame rates (typically ≥90 Hz) ensure fluid interaction and mitigate motion sickness, especially during rapid head or body movements.
- **Interactive Exploration**
  Users can navigate virtual spaces and manipulate content via VR controllers, hand tracking, or voice commands, enhancing engagement and usability.
- **Data Integration**
  Applications often visualize large, dynamic datasets sourced from simulations, sensors, CAD/BIM models, or enterprise systems, requiring robust real-time processing.
- **User Presence**
  A successful VR experience maintains a strong sense of “being there” through responsive tracking, high-fidelity visuals, and synchronized audio-visual feedback.
- **Multi-User Collaboration**
  Advanced solutions support multiple simultaneous users, enabling remote collaboration and shared decision-making within immersive environments.

4.1.1.2 Technical Requirements for VR Visualization

Developing effective VR applications entails meeting several stringent technical requirements:

- **High Computational Power**
  Real-time rendering and data processing demand robust GPU/CPU capabilities, especially for complex geometry or dynamic simulations.
- **Low Latency**
  System responsiveness—measured in milliseconds—is critical to preventing motion sickness and ensure natural interactions.
- **Optimized Data Pipelines**
  Efficient loading, caching, and streaming mechanisms are required to handle large 3D datasets or real-time telemetry.
- **Robust Tracking**
  Precise 6DoF tracking of head and hands is essential for maintaining spatial alignment and interaction fidelity.
- **Ergonomics and Comfort**
  Hardware must support extended use without causing discomfort, fatigue, or strain, especially in professional or training settings.

4.1.1.3 Development Platforms for VR Applications

Multiple platforms are available for building VR visualization applications, each offering distinct benefits. These are presented in Table 1:

Table 1: VR Applications Platforms

Platform	Description	Pros	Cons
Unity[21]	Cross-platform game engine with XR Interaction Toolkit and extensive VR support	Easy to use; strong asset pipeline; community support	Requires optimization for high-performance VR
Unreal Engine[22]	High-end game engine with advanced rendering (Nanite, Lumen), ideal for photorealistic VR	Top-tier graphics; blueprint scripting; built-in VR tools	Steep learning curve; resource-intensive
Native (C++, Vulkan) [30] [31]	Low-level programming for maximal control via OpenGL/Vulkan	Fine-grained control; best performance	High complexity; slower development
WebXR[23]	Browser-based standard for VR/AR experiences	No install needed; cross-platform; accessible	Limited access to hardware features; performance constraints

4.1.2 SDKs, Platforms, and Tools for VR Visualization and Collaboration

This section provides a comprehensive overview of the key software development kits (SDKs), platforms, and frameworks used for developing immersive VR visualization and collaboration applications, with a particular focus on OpenXR [24] and Unity-based solutions. It includes a comparative analysis of major VR development stacks and networking options for real-time multi-user experiences.

4.1.2.1 OpenXR: The Unified Interface for XR Development

OpenXR [24], maintained by the Khronos Group, is an open, royalty-free API standard that simplifies VR and AR development by offering a unified interface to access a broad range of XR

hardware. It is designed to support application portability and reduce platform-specific code dependencies.

Key Benefits of OpenXR

- **Cross-Platform Compatibility**: Enables developers to target multiple XR devices from a single codebase.
- **Hardware Agnostic**: Abstracts hardware-specific implementations to support a wide range of current and future devices.
- **Improved Performance**: Facilitates more direct communication with hardware, improving latency and rendering speed.
- **Ecosystem Growth**: Encourages industry-wide interoperability by aligning hardware vendors and platforms around a common standard.

While many legacy applications still depend on proprietary SDKs (e.g., Oculus SDK, SteamVR SDK), new development trends favour OpenXR due to its enhanced portability and future-proof design.

4.1.2.2 Meta SDK (Oculus Integration): Optimized for Quest Devices

Meta's SDK [25] (formerly Oculus Integration) provides comprehensive access to hardware-specific features on Quest devices (Quest 2, Quest 3, Quest Pro). Used in conjunction with Unity, it offers native support for hand tracking, passthrough, spatial anchors, and other features that enable high-performance applications tailored for Meta hardware.

Capabilities include:

- Integration with Unity for seamless hardware interaction
- Support for hand tracking, eye tracking, haptics, and MR passthrough
- Meta XR Plugin (OpenXR backend) for optimized rendering
- Platform SDK features such as voice services, avatars, and cloud anchors

4.1.2.3 Comparison of Leading VR Development Platforms

Table 2 summarizes the most prominent VR visualization tools and their strengths and limitations:

Table 2: Leading VR Development Platforms

Tool / Solution	Development Platform	Pros	Cons
Unity + OpenXR	Unity Engine	- Cross-platform support- Strong community- Good for multiplayer- GLTF support- XR Interaction Toolkit	- Needs optimization- VR UI requires custom work
Unity + Meta SDK (Oculus Integration)	Unity Engine + Meta SDK	- Native support for Quest hardware- Hand, eye tracking, passthrough- Optimized rendering- XR Interaction Toolkit	- Meta-specific- Quest hardware limits
Unreal Engine 5 + OpenXR	Unreal Engine 5	- High fidelity (Lumen, Nanite)- Blueprint scripting- Pixel Streaming	- Steep learning curve- High system requirements- Complex deployment

NVIDIA Omniverse XR	USD-based, connects to Unity/Unreal/Blender	- Real-time RTX rendering- USD interoperability- Cloud collaboration support	- Requires RTX GPUs- Enterprise-oriented- Learning curve for USD
Varjo Reality Cloud + Base	Varjo Hardware + Varjo Base Software	- Human-eye resolution- Eye/hand tracking- Remote visualization	- Very high cost- Hardware-specific- Not indie-friendly
Blender + VR Scene Inspect	Blender (OpenXR integration)	- Free & open-source- Easy for asset inspection- Lightweight	- Basic VR interactivity- No networking or app logic
Mindesk	Plugin for Rhino, Grasshopper, SolidWorks	- Real-time VR in CAD- Parametric updates- Good for walkthroughs	- Paid software- CAD-specific- Limited interaction options

4.1.2.4 Networking Options for Multiplayer VR

When developing multiplayer VR applications, choosing the appropriate networking solution is critical. Table 3 presents a comparative overview of three major Unity-compatible networking stacks:

Table 3: Networking Options for Multiplayer VR

Feature/Aspect	Mirror	Photon Unity Networking (PUN)	Unity Netcode for GameObjects (NGO)
Type	Open-source, self-hosted, high-level	Commercial, managed cloud	Unity Official, self/relay-hosted, high-level
Cost	Free	Free tier, then paid by CCU	Free core, usage-based for Unity Relay
Server Hosting	Self-hosted	Exit Games managed cloud	Self-hosted + Unity Relay
Scalability	Manual scaling	Excellent (auto-scaled)	Good with Unity Relay/Multiplay
Ease of Use	Good, strong community	Very good, detailed tutorials	Good, Unity-integrated
Development Model	Client-server	Client-server (semi-authoritative)	Server-authoritative recommended
VR Focus	General-purpose	General-purpose	General-purpose
Persistence	Custom implementation	Custom implementation	Custom implementation
Data Sync	RPCs, SyncVars, NetworkTransform	RPCs, PhotonViews, TransformView	RPCs, NetworkVariables, NetworkTransforms
Community	Active open-source	Large official support community	Growing Unity community
Unique Features	Custom transport layer	Fusion (ECS), Realtime low-level API	Client-side prediction, authoritative sync

Summary of Pros and Cons:

- **Mirror**
  - *Pros*: Fully open-source, self-hosted, highly customizable
  - *Cons*: Requires advanced server management; lacks built-in cloud features
- **Photon Unity Networking (PUN)**

- *Pros*: Simplifies deployment with managed cloud, well-supported
- *Cons*: Paid plans for scale, limited server control

- **Unity NGO**
  - *Pros*: Official Unity solution, good integration, supports relay/lobby
  - *Cons*: Newer ecosystem, learning curve for authoritative game models

4.1.2.5 Unity as the Optimal Platform for VR Visualizers

In the context of developing a multiplayer VR visualizer, the choice of Unity over other available solutions—such as Unreal Engine or custom-built platforms—is driven by a combination of factors that effectively meet the specific demands of interactive, real-time virtual reality experiences involving multiple users. Unity often stands out as the most suitable option due to several technical and operational advantages.

First, Unity provides a highly efficient workflow for VR development, with native support for a wide range of headsets through integration with OpenXR and platform-specific SDKs (e.g., Meta Quest, SteamVR, Pico). Tools such as the XR Interaction Toolkit enable rapid implementation of common VR interactions—including object manipulation, teleportation, and UI handling—accelerating development timelines, especially for visualizers where interaction with 3D content is essential. On the multiplayer side, Unity offers robust networking capabilities through Netcode for GameObjects (NGO), which is optimized for server-authoritative, real-time applications. This ensures reliable synchronization of user positions, interactions, and shared environments, which is fundamental in collaborative VR contexts. Additional services like Unity Relay and Lobby further simplify session management and network traversal, facilitating smoother user connections across different network configurations. While Unity NGO is not designed for large-scale MMOs, it can effectively support a substantial number of concurrent users, making it well-suited for collaborative scenarios such as design reviews or walkthroughs in virtual environments. From a visualizer perspective, Unity's asset pipeline is highly optimized for importing and managing complex 3D models from CAD or BIM software—thanks in part to tools like Unity Industry and Unity Reflect—allowing for efficient handling of architectural, engineering, or industrial designs. Furthermore, Unity provides comprehensive profiling and optimization tools to ensure the high and consistent frame rates required for comfortable VR experiences, particularly on standalone devices such as the Meta Quest. The flexibility of Unity's UI system (Canvas) also enables the creation of both traditional and immersive spatial interfaces, which are crucial for intuitive navigation and interaction in VR environments. Unity's real-time engine allows users not only to view models but also to interact with them—changing materials, triggering animations, toggling layers, or collaboratively editing scenes—thereby enhancing the interactivity of the experience.

Lastly, Unity benefits from a vast ecosystem and an active developer community. The Unity Asset Store offers a broad library of VR-specific assets, tools, and networking modules, reducing development time. Its use of C#, a widely adopted and accessible object-oriented language, further lowers the entry barrier for development teams. While alternatives like Unreal Engine may offer superior visual fidelity and are often chosen for cinematic or AAA-grade VR experiences, they tend to have steeper learning curves and higher performance requirements. Custom-built solutions, though potentially powerful, typically demand significant engineering effort and are viable only in highly specialized projects with ample resources. In conclusion, Unity presents a balanced and mature platform for developing multiplayer VR visualizers. Its native VR support, powerful networking tools, optimized asset handling, and strong community support make it a pragmatic and effective choice for building collaborative, immersive environments for design review, training, or product visualization.

4.1.2.6 Recommended Stack: Unity + Meta SDK for Quest

For projects focused on immersive 3D visualization and collaboration on Meta Quest devices, Unity combined with the Meta SDK offers the most balanced solution. Unity provides a flexible development environment with broad industry adoption, while Meta's SDK ensures deep hardware integration, optimized rendering, and access to platform services like voice chat and entitlement checks.

Key advantages include:

- Seamless hand and eye tracking, passthrough MR, and spatial anchors.
- Scalable multiplayer with Unity networking tools (NGO, Relay, Lobby).
- Efficient CAD/BIM import via Unity Industry/Reflect and GLTF support.
- Full control over rendering pipeline, UI/UX, and asset workflows.

This stack is particularly well-suited for applications in design review, education, engineering, and collaborative virtual environments.

4.2 Features & Functionalities

4.2.1.1 XR Client for Unity

The XR Client for Unity (XRCU) is the core module of the TwinEU XR Framework. It is a modular, real-time multi-user framework built on the Unity 3D platform, designed to enable immersive collaboration in both co-located and remote scenarios.

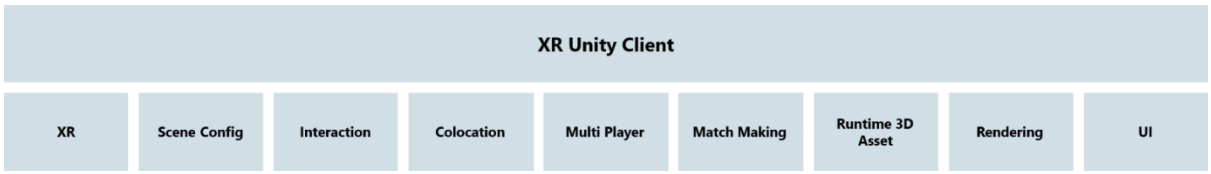

Figure 4: XR Client for Unity

It supports dynamic manipulation and synchronization of digital assets through real-time networking, leveraging open standards such as OpenXR and glTF [26] to ensure cross-platform compatibility and scalability. Targeted primarily at the Meta Quest headset, the XRCU integrates the Meta SDK to establish a shared world origin, enabling seamless colocation experiences while guaranteeing optimal device performance and access to hardware-specific features. The architecture is composed of specialized modules (Figure 4), each responsible for a critical function: user interaction, networking, scene management, asset streaming, input handling, and spatial awareness.

Scene Config

At the core of the setup process is the Scene Config module, which utilizes a JSON-based configuration file to define the XR environment. This includes specifying GLTF models, environmental lighting, interaction behaviour, and multiplayer roles. The use of external JSON allows real-time scene updates without altering application code, enhancing flexibility and maintainability. This configuration is parsed at runtime by the JSONParser class within the XR module, which serves as the central orchestrator.

XR Module

The XR module oversees rendering, multiplayer communication, and runtime asset management, and ensures synchronized updates and interactions across users. The JSONParser extracts structured data from the Scene Config, initiates asset downloads, configures interaction settings, and dynamically assembles UI elements based on the scene context.

Colocation

The Colocation module ensures that multiple users can share the same physical and virtual space, a critical capability for collaborative design, mixed reality training, and multiplayer scenarios. This is achieved through the establishment of a Shared Spatial Reference (Master Anchor), typically implemented via the Meta SDK's shared world origin. Other approaches—such as fiducial markers or cloud-based anchors—are limited by hardware compatibility or deprecated services, making the platform SDK the most viable solution for the first release.

User Interface (UI) and Interaction Module

The User Interface (UI) and Interaction Module are fundamental parts of the XRCU system, shaping how users engage with the virtual environment and the 3D content within it. Designed with flexibility and contextual adaptability in mind, these modules offer an intuitive and immersive user experience, whether the user is navigating a complex engineering model, collaborating with others, or manipulating 3D assets in real time.

At the heart of the UI module is the **dynamic UI system**, which is entirely driven by the scene configuration specified in a JSON file. Instead of hardcoding fixed controls into the application, the UI is generated and adapted at runtime based on the specific requirements of the current scene. This means that if a user loads a model that supports sectioning or exploded views, corresponding tools automatically appear within the interface. Conversely, in simpler scenes, only essential controls are shown—resulting in a clean, streamlined interface that avoids overwhelming the user with unnecessary elements. The UI is built using Unity's native 3D UI system, which allows for optimized rendering performance and seamless integration with the XR environment. It supports both hand tracking and controller-based input, ensuring broad compatibility with different user preferences and hardware capabilities. Menus, toolbars, and interaction panels are presented in a spatially aware manner, meaning they can be placed and interacted with in the 3D space around the user, enhancing accessibility and immersion.

Complementing the UI is the **Interaction System**, which defines how users can manipulate and engage with 3D objects in the environment. Standard interaction features include grabbing, rotating, scaling, and moving objects, all made possible through the Meta SDK's robust input handling for hand and controller tracking. These core interactions are intuitive and responsive, allowing users to manipulate digital assets as if they were physical objects. Beyond basic manipulation, the system also includes **Custom Interactions** tailored for advanced use cases. Features like **Sectioning**, which lets users slice through models to view internal structures, and **Exploded Views**, which break complex assemblies into their component parts, expand the range of XR applications. These tools are especially valuable in training, product design, and engineering, where detailed inspection of objects is required.

To support collaborative use, the interaction system also manages **object ownership**. When a user interacts with an asset, the system assigns temporary ownership, preventing others from manipulating the same object simultaneously. This behaviour is coordinated using Unity's networking framework and is essential for maintaining synchronization and avoiding conflicts in multiplayer sessions.

Multiplayer

The **Multiplayer Module** lies at the core of the collaborative XR experience, enabling multiple users to interact in a shared virtual environment in real time. This module is designed to support both co-presence and synchronized interactions by integrating matchmaking, session management, and real-time communication features. To oversee communication and synchronization, the module relies on **Unity's Netcode for GameObjects (NGO**) [27]. This framework is responsible for managing connections, distributing ownership of objects, and synchronizing data across clients. It supports low-latency communication through direct peer-to-peer links or relay servers, depending on network conditions.

A critical aspect of the multiplayer module is its ability to support **runtime player entry**. New users can join a session that is already in progress, and the system ensures they receive a fully synchronized scene. The integration with the **GLTF Importer** means that any 3D models spawned dynamically during the session are also instantiated and networked for late joiners. Multiplayer interactions—such as grabbing, scaling, or sectioning 3D models—are managed through ownership and synchronization mechanisms. When a player begins interacting with an object, the system assigns temporary ownership using the OwnershipBehaviour logic, ensuring conflict-free manipulation. Meanwhile, the SelectionManager tracks ongoing interactions to maintain a consistent state across all users. In essence, the Multiplayer Module transforms a single-user XR experience into a fully collaborative space, where presence, communication, and interaction are synchronized in real time. Its modular structure, based on industry-standard networking tools and custom XR logic, allows for scalable, stable, and intuitive multi-user scenarios—paving the way for rich, immersive shared experiences.

Matchmaking

A seamless multiplayer experience is essential for immersive XR environments, and the XRCU addresses this need through an integrated matchmaking system that automatically connects users based on a set of predefined conditions. Rather than requiring users to manually select sessions or coordinate externally, the system intelligently manages session discovery, joining, and creation behind the scenes. When a user launches the application, the matchmaking logic begins by searching for active sessions that match specific parameters. These parameters typically include the scene configuration in use, the user's geographical region, and the number of players currently connected to a session. If an existing session meets these criteria, the user is seamlessly connected, without any need for manual input. However, if no compatible session is found, the system does not leave the user stranded. Instead, it automatically creates a new session, or "lobby", which becomes available for others to join. This new lobby can be dynamically discovered by other users who share the same matching criteria. The user who initiates the session assumes the role of host and can define certain configuration aspects of the experience, such as which assets are available on the scene or how user interactions are managed. The underlying networking infrastructure is powered by Unity's NGO, a robust framework that manages the complexities of multiplayer connections. NGO takes care of synchronizing player states, handling connection events, and maintaining consistency across the session. Depending on the quality of the network connection and setup, the system can switch between direct peer-to-peer connections or routing communication through relay servers to ensure the best possible latency and stability. An important feature of this system is its ability to support dynamic player entry. New participants can join an ongoing session at any time. When this happens, the system ensures that all relevant scene data—such as the state and position of objects, avatars, and interactions—is correctly synchronized for the new player. This is made possible through coordination between the matchmaking system and the GLTF Importer, which manages the instantiation and networking of 3D assets, even after the session has already begun.

Runtime 3D Asset

The Runtime 3D Assets module is designed to dynamically load glTF models during runtime, rather than preloading them at application startup. This strategy enhances performance by reducing initial memory usage and allowing assets to be fetched only when needed, resulting in a smoother and more responsive user experience. Many glTF models include nested objects or hierarchical structures. To support multiplayer interactions with these sub-components, the system:

- Reconstructs the object hierarchy upon loading.
- Assigns NetworkBehaviour components to relevant child objects.
- Attaches interaction scripts to enable manipulation of individual parts.

This reconstruction ensures reliable, consistent interactions with complex models, allowing users to engage with both the whole object and its individual components in a fully synchronized, networked environment.

4.2.1.2 XRCU Architecture

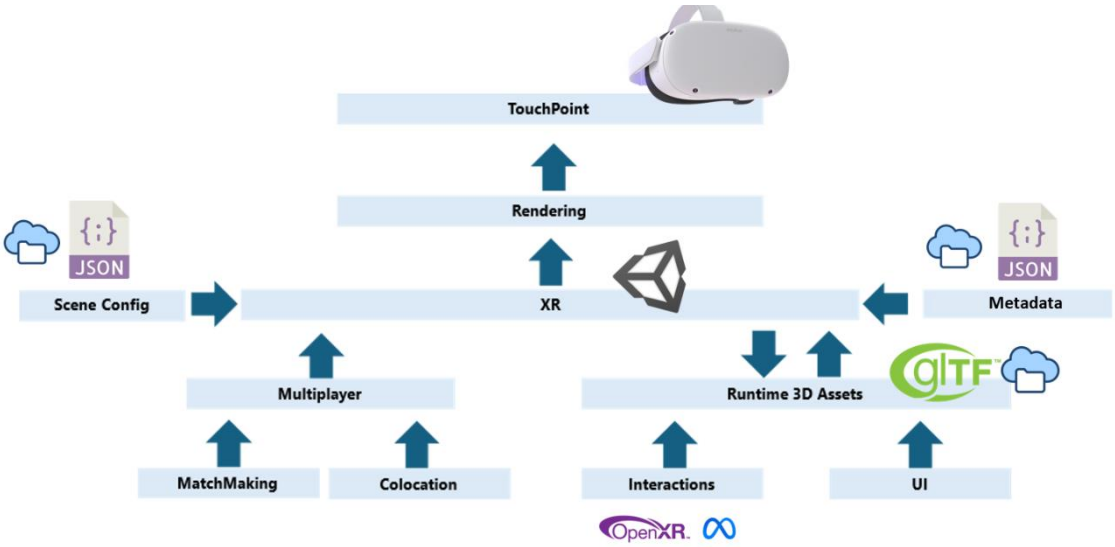

Figure 5: XRCU Architecture

Figure 5 illustrates the high-level structure of the XRCU, highlighting the modular flow and component interaction required to deliver collaborative, immersive XR experience on devices such as the Meta Quest headset. At the centre of the architecture is the **XR Core**, which acts as the main orchestration hub. It integrates and coordinates all modules involved in the XR experience—from rendering and user interaction to multiplayer networking and runtime asset management. On the **input side**, the **Scene Config** module retrieves a JSON-based configuration file from an external repository (cloud). This file defines the setup of the virtual environment, including which 3D models (in glTF format), interactions, lighting, and multiplayer settings should be used. The XR module parses and applies these configurations to initialize the scene. Simultaneously, the **Metadata** module also retrieves scene-specific data from an external JSON source, enriching the runtime environment with additional information linked to 3D assets. The **XR module** drives several key subcomponents:

Multiplayer, which is subdivided into **MatchMaking** (for session management and user connection) and **Colocation** (for aligning multiple users in a shared spatial reference using platform SDKs like Meta's). At its foundation, the module ensures that users can connect to a common session

through the **Matchmaking System**, which automatically groups users based on shared scene configurations and other criteria. This allows for dynamic creation and discovery of multiplayer lobbies, ensuring that users can either join existing sessions or initiate new ones effortlessly. Once a session is established, the **Colocation** component enhances the sense of shared presence by aligning user positions and interactions within a common spatial frame. This is especially important in XR environments, where spatial awareness and consistency are key to immersive collaboration. Colocation ensures that all users perceive each other—and shared objects—in the same virtual space, maintaining a coherent sense of orientation and distance.

Runtime 3D Assets, which manages the loading and synchronization of glTF models and their associated metadata. This is closely tied to the **Interactions** and **UI** modules:

- **Interactions** manage how users manipulate the environment, supporting both standard and advanced techniques like grabbing, rotating, sectioning, and exploding views, with support from platforms like OpenXR and the Meta SDK.
- **UI** dynamically adapts the user interface based on the JSON scene configuration, using Unity's 3D UI system to display context-sensitive tools and menus.

The Rendering module transforms the virtual environment into a visual output suitable for XR devices, and connects directly to the **TouchPoint**, representing the user's view and interaction point through the Meta Quest headset. Overall, this architecture promotes flexibility, real-time configurability, and device-specific optimization by utilizing open standards (OpenXR, glTF) and platform-specific SDKs (Meta SDK). It enables immersive, multi-user experiences with dynamic content and UI, all centrally managed by the XR module for cohesive system behaviour.

4.3 Deployment

The application has been developed as a multiplayer experience, allowing users in the same physical location to interact seamlessly within a shared virtual environment. To evaluate core features such as real-time spatial synchronization, Meta Colocation, and multi-user interactions, it is strongly recommended to evaluate the application with **at least two Meta Quest headsets**. This setup provides a realistic scenario for validating synchronization accuracy, interaction responsiveness, and the overall user experience. If users are **not in the same physical location**, the application still supports multiplayer mode. However, **Meta Colocation will be unavailable**, and synchronization will instead rely on standard network-based methods. A UI notification within the app will inform users that Colocation is not active in their current session. The user second user (client) must use the UI on the left-hand side to select the 'settings' option, accessible via the gear icon. Under '*Developer Options*', he should then select '*Start without Colocation*'. To gain access to the **beta version** of the XRCU and enable Meta Colocation, follow the steps below:

Step-by-Step Beta Testing Procedure

1. **Provide Meta Account Email**
   a. Share the email address associated with your Meta account with the development team.
   b. This email is required to grant beta access through the Meta Developer Dashboard.
2. **Receive Beta Invitation**
   a. After being added as a tester, you will receive an email invitation from Meta.
   b. This email includes a link to join the beta program.
3. **Accept the Beta Invitation**
   a. Open the invitation email and click the link provided.
   b. Log in using your Meta account credentials.

   c. Accept any terms and conditions if prompted.
4. **Enable Beta Access**
   a. Open the Meta Quest mobile app.
   b. Go to **Library > Apps** and locate the application under the **beta** section.
   c. Confirm that the app is listed and accessible for beta testing.
5. **Download the Beta Application**
   a. On your Meta Quest headset, open the Meta Quest Store or navigate to **Library**.
   b. Search for the application or locate it in your app list.
   c. Download and install the beta version.
6. **Grant Spatial Permissions for Meta Colocation**
   a. Meta Colocation requires access to **Spatial Anchors** for shared spatial experiences.
   b. When prompted, grant all necessary permissions for spatial tracking.
   c. Without these permissions, Colocation features will not function properly.

5 Data Space adaptation to support the TwinEU continuum

5.1 TwinEU Digital Twin Federator

Enabling a federated ecosystem for DTs in the energy sector a complex task, but TwinEU project has a clear vision on it: implementing a **Digital Twin Federator** with main goals of:

- Integrating heterogeneous DT systems
- Orchestrating services for data and models sharing
- Incorporating real-world data to enhance simulation and analysis capabilities

To achieve these goals, the TwinEU DT Federator leverage on the Data Space concept for supporting an interoperable and secure DT Federation with easy and effective integration of various data sources and infrastructures. A dataspace-enabled structure for data and model sharing is a pivotal element of the TwinEU architecture. It adapts the Data Space concept to the specific context of TwinEU DT ecosystem, creating a trusted, sovereignty-preserving layer for data and DT model exchange.

5.1.1 OneNet Data Space Framework

The TwinEU DT Federator extends and improves the **OneNet Data Space Framework**, (Figure 6) an open-source Data Space solution tailored for the Energy domain, which support the standardized data exchange among energy stakeholders at any level. At architectural level it includes two main components: the **OneNet Middleware and the OneNet Connector** (Figure 7).

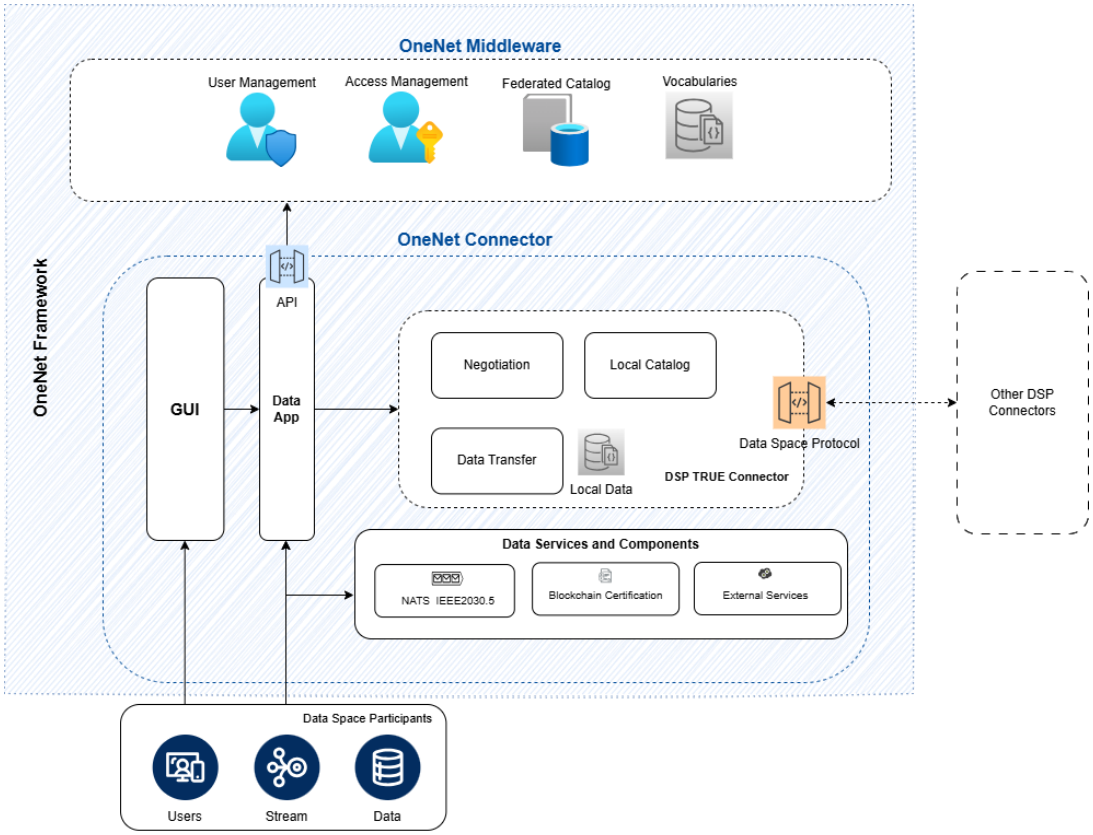

Figure 6: OneNet Data Space Framework

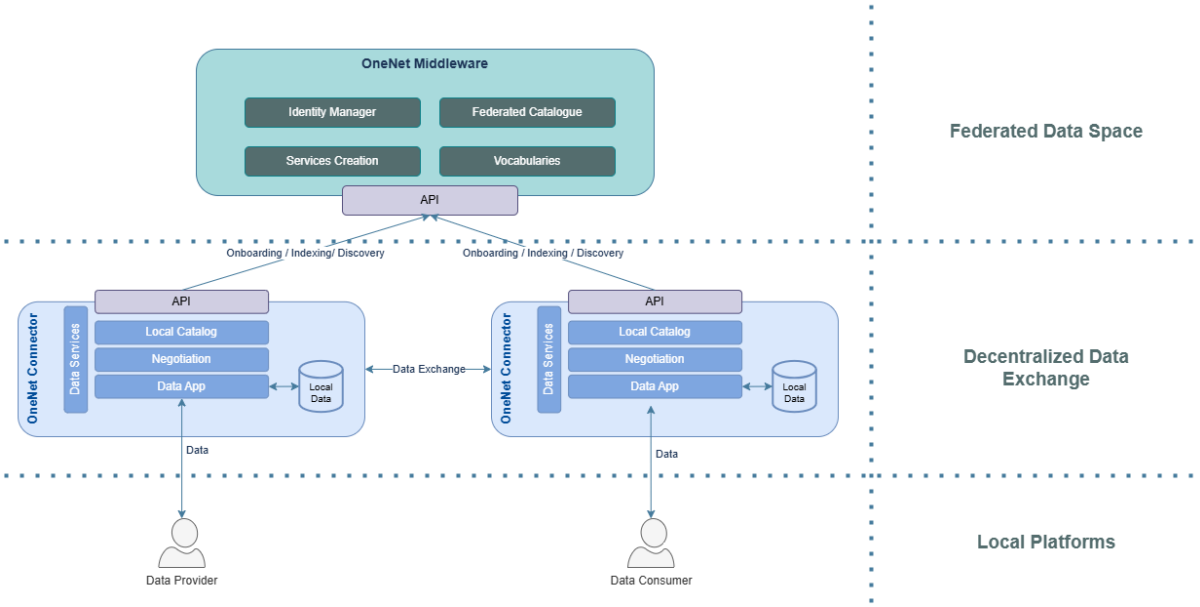

Figure 7: OneNet Connector Architecture

The **OneNet Middleware** stands on top of the common infrastructure, ensuring secure and standardized communication between assets, systems, data sources, models, and energy stakeholders. This middleware facilitates seamless, efficient, and transparent sharing of data across the entire ecosystem, creating the concept of Federated Data Space.

The OneNet **Connector** plays a crucial role in driving efficient data integration and exchange across the energy sector. It implements the core features for a completely decentralized end-to-end data exchange while maintaining full control over data access and usage. In addition, it communicates with Middleware for onboarding and data service discovery. The Connector is the core of the IDS Reference Architecture Model for implementing the concept of Data Space. It serves as the gateway for connect existing systems and their data to an IDS ecosystem. Its architecture and functionalities are defined by the IDS Reference Architecture Model (RAM) and specified by the certification criteria.

The Connector facilitates secure data exchange while enriching it with metadata, including customizable usage conditions that can be defined, managed, and enforced directly within the Connector. A key benefit of this approach is data sovereignty—ensuring organizations retain full control over how their data is accessed and used. The metadata follows the IDS Information Model ontology, maintaining standardized and structured information exchange. The IDS reference architecture, with its decentralized data storage, enables seamless integration of data from various sources while restricting access exclusively through other IDS Connectors. This decentralized and controlled data-sharing approach guarantees that data owners always determine who can use their data, under what conditions, and for what purposes, reinforcing both security and trust in data-driven collaborations.

5.1.2 Evolution from OneNet Framework and First Release

In the context of T4.5, the OneNet Framework had evolved as DT Federator for supporting the DT federation and several additional features such as:

- Technical Interoperability: Data Space Protocol
- Semantic Interoperability: Extension of Vocabulary and Standards
- DT support: Data and Models Exchange

- New technologies, protocols and standards for Real-time data exchange

This first release of the DT Federator, integrated within the TwinEU Platform focused on implementing the Data Space Protocol for ensuring a high level of interoperability in the context of Energy Data Space. All the Data Space Protocol (DSP) modules and interfaces were successfully integrated in a new version of the OneNet Connector for DTs, with a dedicated UI and a new Data Application, which allow the integration of legacy systems. The remaining mentioned features will be implemented in the next period of the project and reported in the coming deliverables.

5.2 Features & Functionalities

The main functionalities of the initial version of the OneNet Connector for DTs, implemented during the first period of the project, can be structured into 5 steps:

1. Configuration (Onboarding)
2. Creation of Data Offering (Indexing)
3. Data Catalogue (Discovery)
4. Subscription and access management (Negotiation)
5. Provisioning and Consuming Data (Data Exchange)

Figure 8 provides a high-level sequence diagram of the five main phases between Data Provider and Data Consumer, showing the integration of the OneNet Middleware and the OneNet Connector.

Configuration (Onboarding)

This initial step consists of setting up the data space environment via the OneNet Connector. It includes configuring the necessary endpoints, defining participant identities and ensuring that all participants are properly onboarded into the Data Space. This step ensures that the data space is secure and ready for use.

Creation of Data Offering (Indexing)

In this step, Data Providers create and index their data offerings. This involves describing the data, setting metadata, and making it discoverable within the OneNet Middleware. Indexing helps in organizing the data so that it can be easily searched and accessed by users through the Data Catalogue.

Data Catalogue (Discovery)

The Data Catalogue is a centralized repository where all indexed data offerings are listed. Users can browse, search, and discover data sets that are relevant to their needs. This step is crucial for facilitating easy access to data and ensuring that users can find the data they need efficiently. Within the OneNet Middleware, the Data Catalogue also provides categorization mechanisms and a set of vocabularies to be applied to any data offering. The Data Catalogue is implemented in a standardized way, following the Data Space Protocol specifications. The DSP indicate the Catalogue as a collection of entries representing Datasets and their Offers that is advertised by a Provider Participant. The DSP Catalogue uses the standard DCAT model [28].

Subscription and access management (Negotiation)

Once Data Consumers find the data they need, they can subscribe to the data offerings. This step involves negotiating access terms, such as usage rights, and data sharing agreements. Effective subscription and access management ensure that data is shared securely and in compliance with agreed-upon terms. The negotiation step is implemented in accordance with the Data Space Protocol negotiation steps and workflow, which includes six distinct steps, as shown in Figure 9 below.

Provisioning and Consuming Data (Data Transfer)

In the final step, the subscribed data is provisioned to the users. This involves the actual transfer of data from the provider to the consumer, ensuring seamless and efficient data exchange in an end-to-end manner, without passing data through any central system. In the DSP protocol the Data Transfer can be implemented in two-way approach:

- **Push process** – provider push data on consumer service
- **Pull process** – consumer request data to provide

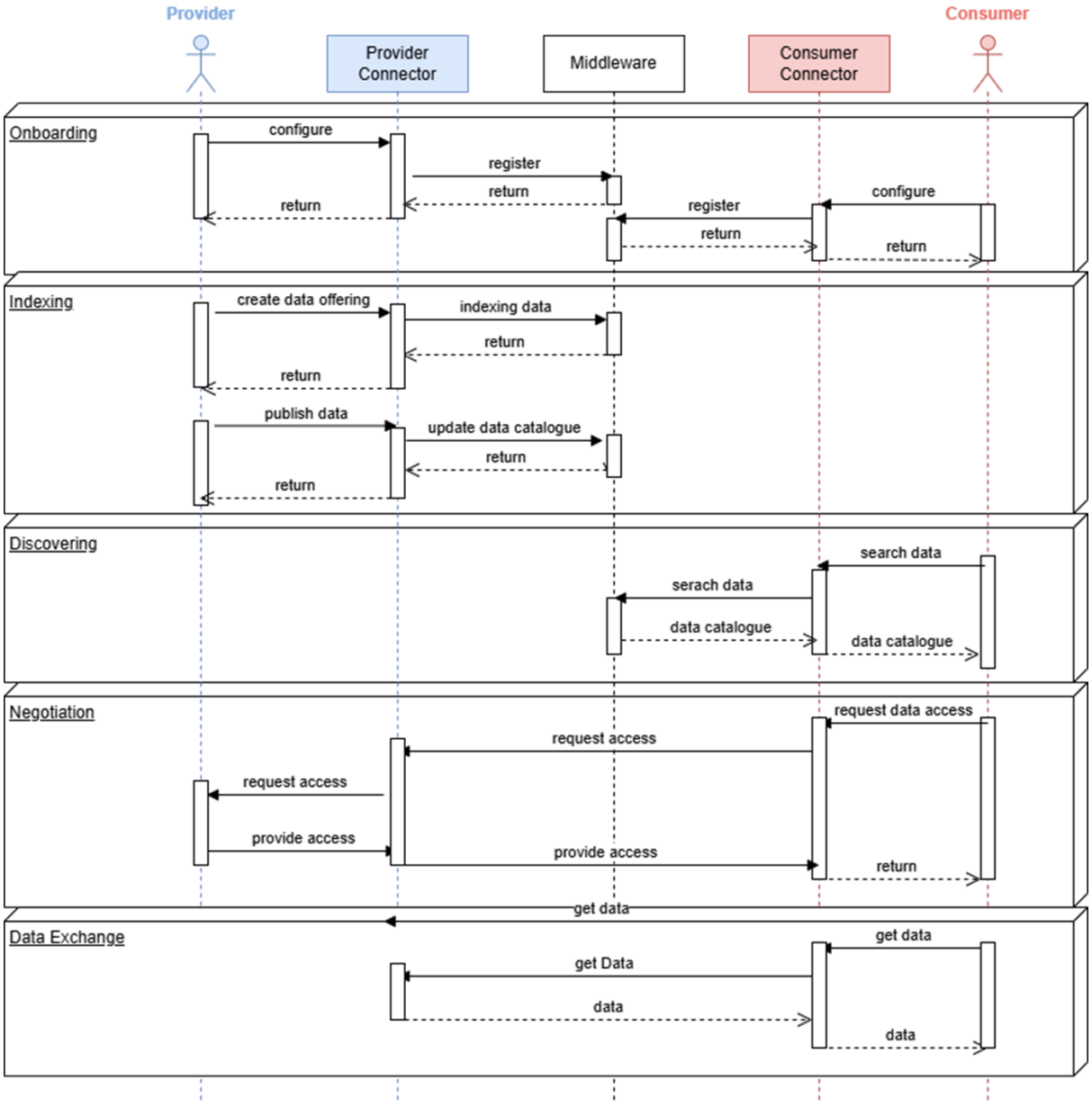

Figure 8: TwinEU Data Space Business Process

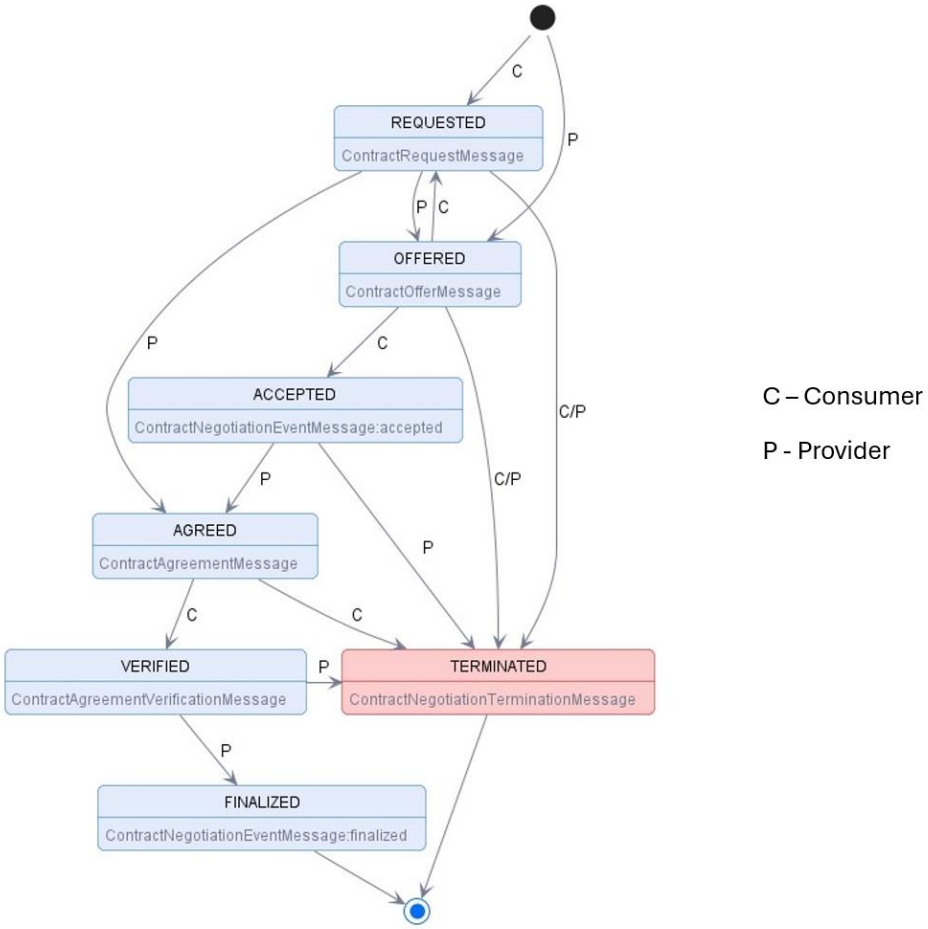

Figure 9: Data Space Protocol Negotiation Steps [29]

5.2.1 Interfaces – REST APIs and GUI

5.2.1.1 Connector APIs

Table 4: Configuration and Onboarding

Name	URL	Method
User Authentication	/api/user/auth	POST
User Info	/api/user/current	GET
Get Connector Settings	/api/custom-query/data-objects/?id=e48046c9-0b94-41d2-9ad4-206f1604b821	GET
Save Connector Settings	/api/custom-query/data-objects/?id=e48046c9-0b94-41d2-9ad4-206f1604b821	POST

Table 5: Data Indexing and Catalogue

Name	URL	Method
Cross Platform Service List	/api/datalist/cross_platform_service/page//{{page-number}}	GET

Cross Platform Service Get by id	api/dataset/cross_platform_service/{{entity_id}}	GET
Offered Services List	/api/datalist/my_offered_services/page/{{page-number}}	GET
Push Offered Services List	/api/datalist/my_push_offered_services/page/{{page-number}}	GET
Offered Services Get By Id	/api/dataset/my_offered_services/{{entity_id}}	GET
Offered Services Create/Update	/api/dataset/my_offered_services	POST
Subscription List	/api/datalist/my_subscriptions/page/{{page-number}}	GET
Push Subscription List	/api/datalist/my_push_sub/page/{{page-number}}	GET
Subscription Get By Id	/api/dataset/my_subscriptions/{{entity_id}}	GET
Subscription Create	/api/dataset/my_subscriptions	POST
Request List	/api/datalist/requests_on_offered_services/page/{{page-number}}	GET
Push Request List	/api/datalist/push_subscription/page/{{page-number}}	GET
Request Get By Id	/api/dataset/requests_on_offered_services/{{entity_id}}	GET
Request Create	/api/dataset/requests_on_offered_services	POST

Table 6: Negotiation and Data Transfer

Name	URL	Method
Data Provided List	/api/datalist/data_provided/page//{{page-number}}	GET
Data Provided Get By Id	/api/dataset/data_provided/{{entity_id}}	GET
Data Provided Create	/api/dataset/data_provided	POST
Data Consumed	/api/datalist/data_consumed/{{page-number}}	GET
Push Data Consumed	/api/datalist/push_data_consumed/{{page-number}}	GET
Data Consumed Get By Id	/api/dataset/data_consumed/{{entity_id}}	GET
Timeline List	/api/timeline/data/?id=d6342c52-8995-4a0d-b42d-894ffc600a3d&enabled=1	GET

5.2.1.2 Data Space Protocol APIs

Table 7: Catalogue

Name	URL	Method
Add Catalog	/api/v1/catalogs	POST
Get Catalog	/api/v1/catalogs/{{catalog_id}}	GET
Update Catalog	/api/v1/catalogs/{{catalog_id}}	PUT
Delete Catalog	/api/v1/catalogs/{{catalog_id}}	DEL
Get All Catalog	/api/v1/catalogs/	GET

Table 8: Negotiation

Name	URL	Method
[C] Start negotiation	/api/v1/negotiations	POST
[P]Find Contract Negotiation	/api/v1/negotiations?role=provider	GET
[P] Approve negotiation	/api/v1/negotiations /{{contract_id}}/approve	PUT
[C] Verify negotiation	/api/v1/negotiations /{{contract_id}}/verify	PUT
[P] Finalize negotiation	/api/v1/negotiations /{{contract_id}}/finalize	PUT

[C/P] Terminate negotiation	/api/v1/negotiations /{{contract_id}}/terminate	PUT
[C] Get agreement id	/api/v1/negotiations?role=consumer	GET

Table 9: Data Transfer

Name	**URL**	**Method**
[C] Find Initialized Transfer Process	/api/v1/transfers?state=INITIALIZED&role=consumer	GET
[C] Request transfer	/api/v1/transfers	POST
[P] Find Transfer Process	/api/v1/transfers?role=provider	GET
[P] Start transfer	/api/v1/transfers{{transfer_id}}/start	PUT
[C] Download data	/api/v1/transfers{{transfer_id}}/download	GET
[C] View data	/api/v1/transfers{{transfer_id}}/view	GET
[C] Complete transfer	/api/v1/transfers{{transfer_id}}/start	PUT
[P] Suspend transfer	/api/v1/transfers{{transfer_id}}/suspend	PUT
[P] Terminate transfer	/api/v1/transfers{{transfer_id}}/terminate	PUT

5.2.1.3 Graphical User Interface (GUI)

Login

The Login page allows users to access to the OneNet connector from the local system. The module is connected to the OneNet Middleware for sending the credentials and authenticate the connector in the Data Space instance.

Figure 10: Connecter Login Page

Data Catalogue

This section displays the Catalogue, which consists of a list of offered services from other users. The Offering are categorized and described using metadata. It is possible to find offerings filtering and querying them with several parameters.

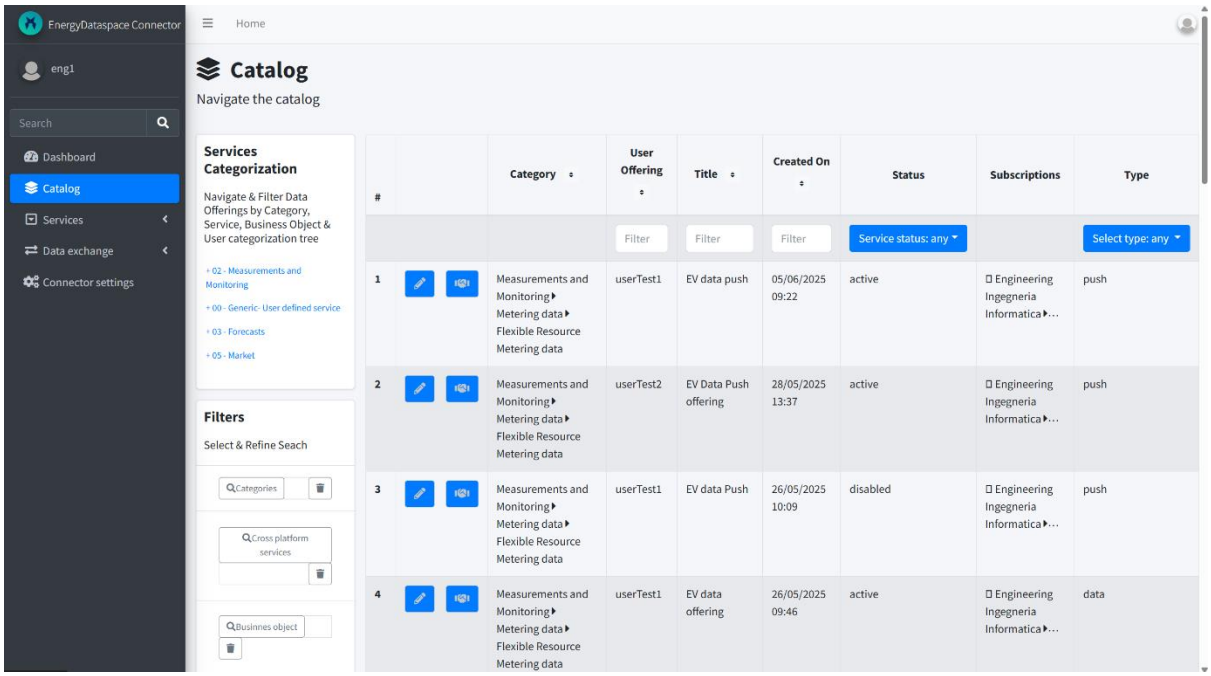

Figure 11: Data Catalogue

Create Data Offering

The services section list shows all services created by the user. In this section is possible to create new Data Offering and publish it into the Data Catalogue.

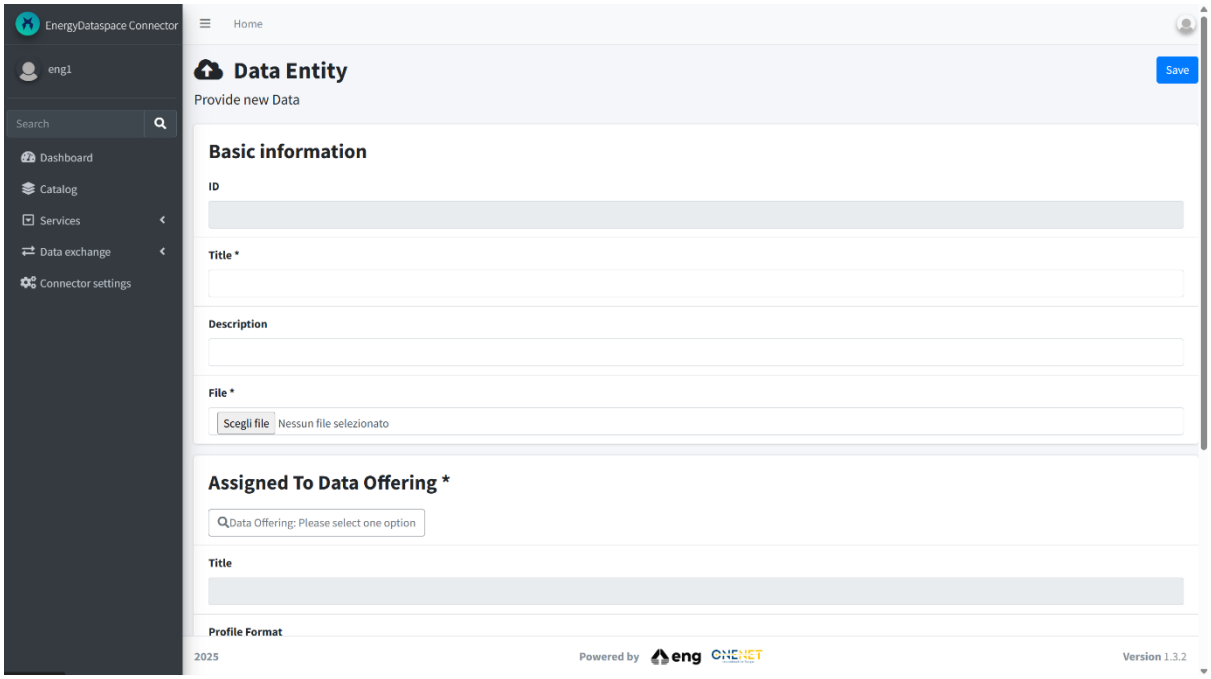

Figure 12: Create Data Offering

Subscription

From the subscription section it is possible to request access to a specific Data Offering from the Catalogue. Once the request from the Consumer's OneNet Connector is submitted, a negotiation phase with the Provider begins.

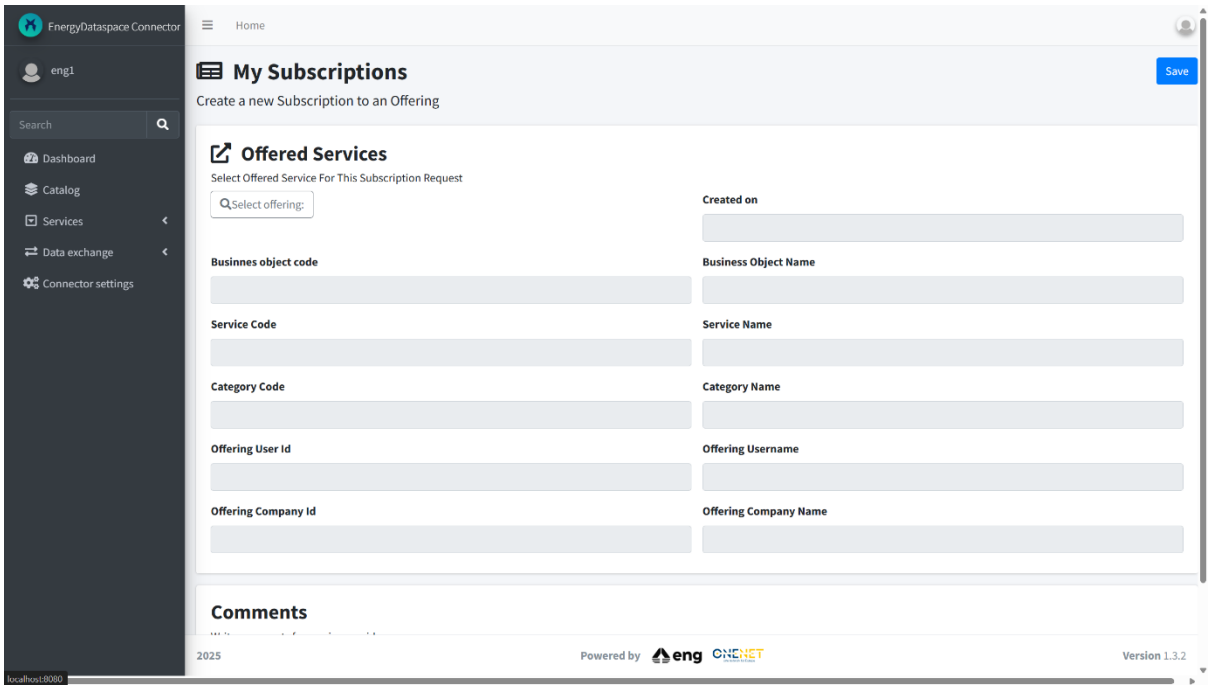

Figure 13: Subscribe to Offerings

Data consuming

Once a data request is approved and the negotiation phase is completed, the Consumer is authorized to access and consume the data. It could also be done directly from the dedicated section of the OneNet Connector GUI.

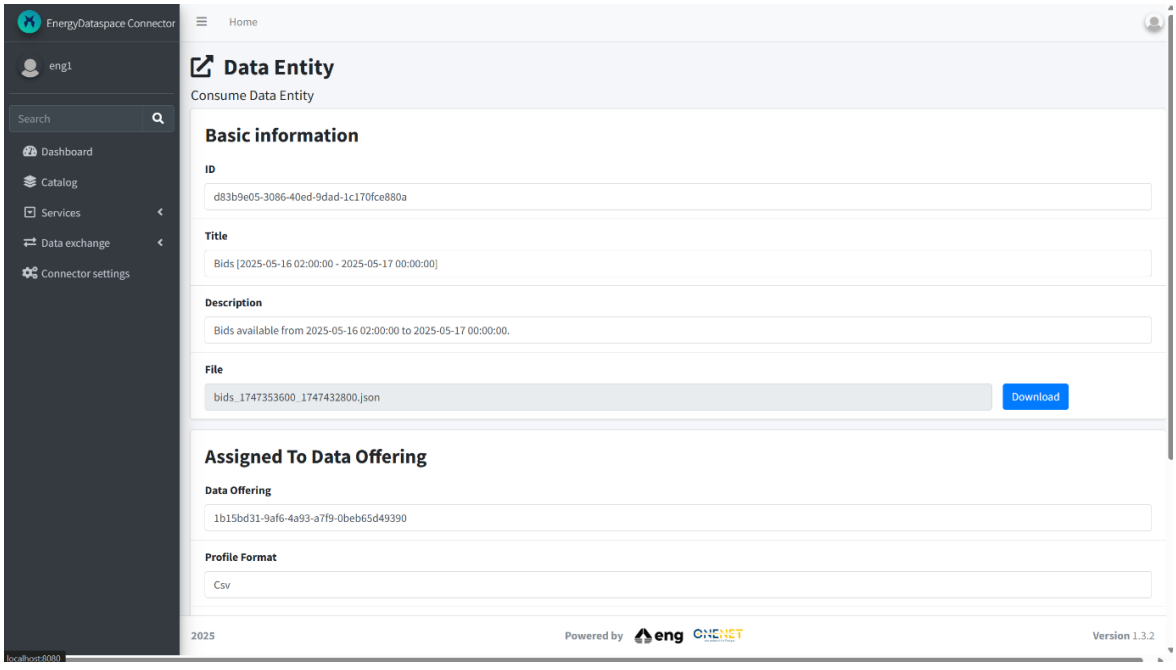

Figure 14: Consume Data

5.3 Deployment

A complete documentation for deploying and installing the OneNet Connector is provided in the TwinEU repository: https://github.com/TwinEU-Digital-Twin-for-Europe/data-space-connector.

The deployment process involves the use of Docker containers. The use of Docker guarantees not only an easy deployment process and total portability of the solution, but also a high level of scalability of the released applications.

The hardware and operating system prerequisites are:

1. A 64bit 2-core processor
2. 8GB RAM Memory
3. 50GB of disk space or more

The software prerequisites include:

1. Linux or Windows (preferably Server edition) Operative System (OS)
2. Docker and docker-compose;

OneNet Connector and its modules will be delivered utilizing the Docker containers functionalities. Firstly, the Docker platform must be downloaded and installed accordingly to the OS of the server to host the deployment. For the correct installation of docker and docker-compose, please refer to the official guides: https://docs.docker.com/get-docker/

5.3.1 Docker Installation

To proceed with the installation of OneNet Connector, the user must use the docker folder of the GitHub repository that contains all the necessary configuration.

1. The first step is to clone the code from TwinEU repository a specific folder *energy-data-space-connector*, by typing:

   mkdir energy-data-space-connector

   cd energy-data-space-connector

   git clone https://github.com/TwinEU-Digital-Twin-for-Europe/data-space-connector
2. There is the *docker-compose.yml* file located under the docker folder that contains all the configuration of the Energy Data Space Connector containers. Go to that file by typing the command:

   cd energy-data-space-connector/docker
3. Start the containers with the below commands:

   docker-compose up -d --build
4. To show logs use the command:

   docker-compose logs -f
5. If no errors appear in the logs, the OneNet Connector has been successfully deployed on your premises.

6 TwinEU platform development and integration

6.1 Introduction

The TwinEU Framework comprises three core horizontal layers, as defined in the TwinEU Architecture [18], along with the Cybersecurity and Data Privacy that spans all three. The primary objective of the TwinEU Framework is to deliver a modular architecture that facilitates secure, scalable, and flexible interoperability among TwinEU participants. These modules provide essential functionalities, including authenticated data access, establishment of secure communication channels between participants, and interaction with a centralized registry containing metadata on available datasets and services. Furthermore, the framework enables efficient discovery of data and services and provides an intuitive user interface for system monitoring, configuration, and operational management. The three layers of TwinEU Framework include:

- **TwinEU Middleware –** Serves as the core Integration platform. For the scope of this deliverable the TwinEU Middleware and platform.
- **TwinEU Federation of DTs** - Forms the Data space ecosystem.
- **TwinEU Service Workbench, Augmented Reality visualisation and AI & Data Marketplace**.

At the core of the TwinEU system lies the TwinEU Middleware, which comprises of several crucial components and functionalities detailed in the following section. The Decentralized Middleware acts as a coordination layer that manages peer-to-peer data exchange among TwinEU stakeholders. Each TwinEU stakeholder interacts with the TwinEU Middleware via its User Interface (Connector GUI) and communicates with other participants through the OneNet Connector. This Connector manages connectivity and data services, operating as an instance of the Decentralised Middleware and plays a central role in establishing decentralized, peer-to-peer connections, facilitating data transfers between providers and consumers. The OneNet Connector is the cornerstone of the TwinEU Federation of Digital Twins since it enables the communication between them. Generally, the TwinEU Framework operates as a central hub where participants (and DTs) can register themselves, their services, and their data. This facilitates secure and trusted discovery and access by other authorized entities within the ecosystem. Finally, the TwinEU Framework also incorporates the TwinEU Service Workbench, which supports the registration, validation and testing of new services. This chapter analyses the TwinEU Middleware/ platform, describing its interfaces, functionality, components and its relationship with the OneNet Connector.

6.2 TwinEU Decentralised Middleware Components

The TwinEU Decentralised Middleware is a core and essential component of the TwinEU Framework. Its principal function is to perform robust authentication of TwinEU participants, thereby ensuring adherence to established data access control policies and privacy regulations. Serving as the primary access interface to the broader TwinEU Framework, the middleware also facilitates the establishment of secure, peer-to-peer communication channels among participants. It enables the technical configuration of each participant's TwinEU Connector, including parameters such as metadata broker endpoint settings and related connectivity specifications. Finally, the TwinEU Middleware contains the administrative components that manage and coordinate the interoperability, sovereignty and authorization within the IDSA network. In more detail the middleware contains the following components that are responsible for specific functionalities:

- **The Clearing House** that serves as a trusted third-party component that logs data transactions between participants to ensure transparency, accountability, and compliance with contractual agreements. It provides verifiable audit trails for data exchanges, enabling secure and tamper-proof evidence of all interactions within the data-sharing ecosystem.
- **The Data and Service Catalogue** that operates as a centralized registry where participants can publish metadata describing their available datasets and services. It facilitates efficient discovery and selection of data assets and capabilities by other participants, thereby supporting dynamic and interoperable data exchange across the ecosystem.
- **The Metadata Broker** that acts as an intermediary that exposes and manages metadata related to available data offerings and services from TwinEU participants. It enables participants to filter, search, and retrieve information about published resources, thus supporting semantic interoperability and efficient data discovery within the ecosystem.
- **The Data access and Policies Service** is responsible for validating the authentication of data transfers and verifying that the direction of transmission complies with defined access policies. It operates in accordance with IDSA specifications and interfaces directly with the TwinEU Middleware. Additionally, it ensures that the appropriate data services are active and correctly configured on the sender's Connector to satisfy the operational and policy requirements of the receiving participant. As a prerequisite for any data exchange, this service is executed first during transmission initialization.
- **Semantic Interoperability Enablers** when ontology and semantic configurations are available, act as data format guidelines, to enhance semantic clarity and interoperability among TwinEU Participants. This contributes to overall data harmonization.
- **The Configuration Tool** ensures that every participant's tool (Connector, service or platform) will be properly integrated, registered and configured within the TwinEU Middleware. The Middleware offers a graphical interface (Connector GUI) that serves to set the correct endpoints of the Connector to establish connectivity, software updates and health monitoring.

Additionally, the TwinEU Middleware operates as an administrative tool that configures manages and enables the operation of the OneNet connector. The Connector is the primary interoperability component of the TwinEU System. Each TwinEU Participant deploys a dedicated instance of the Connector, which operates as the gateway between the participant's local environment and the overarching TwinEU infrastructure, including the TwinEU Framework. This component is responsible for executing a set of data services that collectively ensure secure data exchange, enforce data access policies, validate data formatting, perform semantic enrichment, and maintain data quality in accordance with system standards.

6.3 Interfaces & Functionalities

This section outlines the key functionalities and interfaces of the TwinEU Decentralised Middleware, covering internal and external component integration. Additionally, this section describes the functionalities of the TwinEU Middleware that are related the OneNet Connector. The TwinEU Middleware contains crucial components for user identification, authorization and data access and usage policies. The TwinEU Middleware distinguishes the following functionalities and the corresponding interfaces.

- The Middleware manages the TwinEU participants registration and provides a secure list of the already registered participants.
- The Middleware manages the access, identities and authorization of all participants.

- The Middleware could potentially deploy an instance of the Clearing House. All the data exchanged in the connector could be tracked through the Clearing House service.
- The Middleware deploys the Data Offerings Catalogue. Through this component the Middleware manages and updates a data source catalogue.
- The Middleware provides an explorer for all the standard vocabularies and semantic standards. The components with this specific functionality are categorized as part of the Semantic Interoperability enablers.

Additionally, the TwinEU Middleware provides the following interfaces in order to perform its functionality. Some of these interfaces include interfaces related to the connector. The connector should be deployed by each TwinEU stakeholder and installed in the TwinEU Middleware. Furthermore the Middleware is responsible to offer administrative and authorization services to the Connector operation. The aforementioned interfaces are the following:

- Interface between data provider's local connector and the TwinEU Middleware for metadata exchange.
- Interface between TwinEU middleware and the data provider's local connector authentication policy and access.
- Registration of data information availability through an interface between local connector and the context broker for metadata exchange. The Connector must provide a UI for the configuration of the URIs.
- Discovery and request of data from the TwinEU data ecosystem through an interface between the local connector and the broker for metadata exchange.
- Interface between local connectors for metadata exchange between a connector that operates as data provider and one that operates as data consumer.
- Interface between data consumer's local connector and third-party service or platforms within the TwinEU ecosystem regarding access for data exchange/consuming data. The Connector is responsible to register an external platform using REST APIs. Additionally, the Connector is capable to identifying an external platform using REST APIs.
- Connection between data provider's local connector and the TwinEU Middleware for setup of authentication policy and data exchange configuration prerequisites.
- Interface between data provider's local connector and the TwinEU Middleware for metadata exchange.

The above Middleware functionalities (along with the connector GUI) are built upon a specific tool which offers flexible and effective ways to customize and adjust functionality and user interface dynamically, according to specific needs and requirements. This component is named **SOFIA** and allows for the creation of GUIs and services through a user-friendly environment. The functionalities and features that this administrative middleware offers to the TwinEU could be summarized as follows:

- Creation of custom database schemas, tables and data relationships.
- Create, customize and adjust dashboards, menus, forms and lists.
- Create and customize users, roles and access controls. Access control functionality through the configuration resource can assign access policies per user role, ensuring that the user's data is always secure and safe.

A more descriptive presentation of the features of SOFIA and a representative use case of SOFIA can be found in the following link: https://github.com/european-dynamics-rnd/sofia?search=1 .

6.4 Integration Guidelines of the TwinEU Middleware

6.4.1 Prerequisites and Installation

To ensure a stable and efficient deployment of the TwinEU middleware component, the following hardware, operating system, and software prerequisites must be met. These requirements are intended to support the containerized architecture of TwinEU middleware, which is deployed using Docker-based environments to simplify deployment, scaling, and updates.

6.4.2Hardware Requirements

The minimum hardware specifications for a server intended to host TwinEU middleware are:

- **Processor:** Dual-core CPU (2 physical cores) or higher.
  Recommended: Quad-core for improved performance under moderate-to-heavy loads.
- **Memory (RAM):** 4 GB minimum.
  Recommended: 8 GB or more for better handling of multiple services and concurrent operations.
- **Storage:** At least 50 GB of free disk space.
  Recommended: 100 GB or more, especially when dealing with large datasets, logs, or service updates.

6.4.3Operating System Requirements

TwinEU middleware supports deployment on the following server operating systems:

- CentOS 7 (64-bit)
- Windows Server (latest supported versions)

Notes:

- For CentOS systems, ensure SELinux is properly configured or disabled if necessary for Docker.
- System administrators should apply the latest security patches and updates before installation.

6.4.4Software Requirements

In addition to the OS, the following software components must be installed and correctly configured:

- **Docker:** Required for running TwinEU middleware services as containerized applications. Install the latest stable version compatible with your operating system.
- **Docker Compose:** Used to define and run multi-container Docker applications. This is essential for orchestrating the deployment of various TwinEU middleware modules and their dependencies.

6.4.5 Software Deployment Instructions

The TwinEU stakeholder should satisfy the requirements and subsequently follow the specific instructions for the deployment of the TwinEU Middleware that are presented in the following link:

https://github.com/european-dynamics-rnd/OneNet/tree/main

After that the user should deploy the OneNet connector (see specific deployment guidelines above) and configure the connection settings it through the connector user interface. Specifically, the user should navigate to the connector settings through the User Interface menu and define the URLs of the

Local Api Url, Data App Url, Broker Url and Ecc Url. Those four connector applications are running on the containers that the user should have already installed, so the URLs must be configured accordingly in a specific way that is described in the following link:

https://github.com/TwinEU-Digital-Twin-for-Europe/data-space-connector

6.5 Testing Scenarios Methodology

At this stage (M18 of the project) the definition of testing scenarios analyses the principal attributes that should be satisfied by the TwinEU system. The tests focus on verifying connectivity, security, functionality, and compliance of the overall system architecture, ensuring that the middleware and connector components operate as expected before moving to more advanced integration or scalability tests. Specifically, the testing instances that are crucial for the first technological version and implementation are the following:

- **Environment Validation** - Ensures that all the containers and variables are installed and running properly in the set of the desired user's environments without, errors, runtime dependencies, false loaded environment variables and configuration files.
- **Middleware Health and Connectivity** - Ensures that the initial Middleware version is available and accepts connections. This process includes the testing of internal APIs, attempt basic registration of users or test the response formats.
- **Authentication & Authorization -** Checks that verifies security mechanisms are working for participant access. This process includes attempts of unauthorized access to a secured endpoint, token-based authentication via OAuth2/IDS Identity Provider and scenarios that check that the middleware enforces data access policies correctly.
- **Policy Enforcement Test** - Secures that all the users obey to the predefined usage control policies regarding the publishing, offering, consuming and deleting of data and services within the TwinEU Platform.
- **Connector Communication** - Checks a minimal data exchange between two Connectors. This step includes firstly the simulation of a contract request/subscription between two connectors and subsequently the process of connector-to-connector file exchange.
- **Service and Data Registration** - Checks that validated whether services and data offerings can be configured and shared through the Middleware and Connectors. That stage includes the registration of dummy services or data, validation scenarios of the discovery mechanisms and the minimal sharing of metadata between two sides.
- **Logging** – A check that ensures that there is a functional operational observability using functional logs of participants activity within the Middleware and connector services.
- **Error Handling and Recovery** - Validates the resilience of the system to common errors and the proper handling of them. The system should be capable of handle situations such as exceptions, timeouts and fallback errors.
- **Compliance check with IDSA Guidelines** – Validates that the TwinEU architecture aligns with usage control, message formats and metadata of the IDS standards.

The testing methodologies for the following development versions of the Middleware will be presented in the deliverables that describe these future versions.

7 Integration & Development Plan of the TwinEU framework

7.1 Overview of the Development Plan

The following development plan serves as an agile comprehensive guideline intended to support development teams in achieving seamless integration of system components and services. This plan is dynamic in nature, designed to remain flexible and adaptable in response to evolving requirements and challenges encountered throughout the project lifecycle. The implementation activities and integration processes are structured into three distinct development phases, aligned with the overall project timeline. These phases are detailed in Table 10, Table 11, and Table 12, respectively.

Table 10: 1st Development Phase

	Name	WP / Task	Start	End
A	**First Development Phase**	WP3, WP4	M1	M18
A.1	**Definition of Tech Release Components**	WP3, WP4	M1	M12
A.2	**Technical and Functional Requirements definition & review**	WP3	M1	M12
A2.1	Assessment of demo specific services and functionalities	T3.2	M1	M12
A2.2	Analysis of requirements for DT federation concept	T3.3	M1	M12
A2.3	Analysis of requirements for TwinEU Federated DT Dataspace	T3.3	M1	M12
A2.4	Analysis of requirements for Cybersecurity and data privacy	T3.4	M3	M12
A.3	**Reference Architecture Design**	T3.1	M1	M12
MS4	**Availability of TwinEU open reference architecture**	WP3		M12
A.4	**Extrapolation of Functional Specifications**	WP2, WP3	M2	M14
A.5	**Technical Requirements Specification**	WP2, WP3	M2	M14
A.6	**Advanced service Workbench / Big Data / AI Marketplace – First Version**	T4.1	M4	M12
A.7	**Multi-user XR development platform – First Version**	T4.2	M4	M12
A.8	**Middleware development - First version**	T4.4	M6	M14
A.8.1	Base connector implementation	T4.3, T4.4, T4.5	M6	M14
A.8.2	User Interface base version	T4.4, T4.6	M6	M14
A.8.3	Interfaces & APIs	T4.3, T4.4	M6	M14
A.8.4	Base Middleware & Data space components integration	T4.4, T4.5		
A.9	**1st technology development cycle completion**	WP3-4		M18
MS5	Availability of 1st version of TwinEU platform. Integrated platform (Services Workbench, interactive augmented exploration, middleware, integrated solution and validation)	WP4		M18

D4.1	Integration and deployment of TwinEU platform	WP4		M18
A.10	**Coordination with demos and data sharing strategies – First cycle of coordination**	T4.3	M6	M18
A.11	**EU Data Space adaptation to support the TwinEU continuum**	T4.5	M9	M18
A.11.1	First design of the new features of the OneNet Connector	T4.5	M6	M18
A.11.2	First analysis of interoperability requirements, data integration and homogenization	T4.5	M6	M18
D4.2	**TwinEU Open Data Space v1**	T4.5		M18
A.12	**Initial functional validation & testing**	WP3-4	M16	M18
A.13	**Integration and testing -1-**	WP3-4	M16	M18
A.13.1	TwinEU Deployment Guide [Integration Guide update]	WP3-4	M16	M18
A.13.2	Initial Lab Testing	WP3-4	M16	M18
A.13.3	Ensure functional requirements compliance	WP3-4	M16	M18

Table 11: 2nd Development Phase

B	**Intermediate Development Phase**			M24
B.1	**Reference Architecture Design fine-tuning and finalization**	T3.1	M12	M18
B.1.1	TwinEU components finalization	T3.1	M12	M18
B.2	**Cybersecurity and data privacy requirements first version**	T3.4	M14	M20
D3.1	TwinEU open reference architecture	T3.1		M18
B.3	**Functional Specifications Definition final version**	T3.2	M14	M20
B.3.1	Functional requirements middleware evolution	T3.1, T3.2	M14	M20
B.3.2	Integration and Homogenisation final functional requirements	T3.2	M14	M20
B.3.3	Analysis and finalization of the Demos use cases of the DTs	WP2, T3.2	M14	M20
B.3.4	Extract the final functional requirements version from the use cases	WP2, T3.2	M14	M20
B.3.5	User Interface fine-tuning correspondingly with the functional requirements	T3.2, WP4	M14	M20
B.3.6	Integration and Homogenisation sub-layer implementation - Final version	T3.2, WP4	M14	M20
B.3.7	Finalisation of Functional Specifications	T3.2	M14	M20
B.3.8	Concluding Contingency plan & Risk management	T3.2	M14	M20
B.4	**Technical Specifications Definition final version**	T3.3	M14	M20
B.4.1	Finalization of the open and interoperable Federated DT concept	T3.3	M14	M20
B.4.2	Finalization of the adaptive DT concept	T3.3	M14	M20
B.4.3	Analysis of the interoperability requirements	T3.3, T4.1, T4.3, T4.4, T4.5	M14	M20

B.5	**Implementation of advanced service Workbench / Big Data / AI Marketplace – Final Version**	T4.1	M18	M24
B.5.1	Compliance with security policies	T3.4, T4.1	M18	M22
B.5.2	Publishment of the first mature version of the set of services	T4.1	M18	M22
B.5.3	Integration with the Data Space Federated Catalogue	T4.1	M18	M24
B.6	**Implementation of interactive augmented exploration of TwinEU DT – Final Version**	T4.2	M18	M24
B.7	**Middleware development - Intermediate version**		M18	M24
B.7.1	Intermediate connector implementation	T4.3, T4.4, T4.5	M18	M24
B.7.2	User Interface Finalization	T4.4, T4.6	M18	M24
B.7.3	Update of interfaces & APIs	T4.3, T4.4	M18	M24
B.7.4	Authentication & Authorisation base version	T3.4, T4.4	M18	M24
B.7.5	Tools for Legal, Regulatory, Privacy and Cybersecurity Compliance first version implementation	T3.4, T4.4	M18	M24
B.7.6	Integration and Homogenisation sub-layer implementation	T4.4, T4.5	M18	M24
B.7.7	Data quality & harmonisation services	T4.4, T4.5	M18	M24
D3.2	Functional, and technical Specifications	T3.2, T3.3, T3.4		M20
B.6	**Coordination with demos and data sharing strategies – Second cycle of coordination**	T4.3	M18	M24
B.8	**EU Data Space adaptation to support the TwinEU continuum**	T4.5	M18	M24
B.8.1	Second design of the new features of the OneNet Connector to serve the DTs	T4.5	M18	M24
B.8.2	Finalisation of the interoperability requirements, data integration and homogenization	T4.5	M18	M24
B.8.3	Mapping Data Models into NGSI-LD – First version	T4.5	M18	M24
B.8.4	Integration with Middleware	T4.4, T4.5	M18	M24
B.9	**Pre-integration testing**	WP4, WP5-8	M20	M24
B.10	**1st & 2nd Release integration**	WP4, WP5-8	M20	M24
B.11	**Ensure reference architecture compliance**	WP4	M20	M24
B.12	**Post-integration testing**	WP4, WP5-8	M20	M24
B.13	**Pilots Operation Testing**	WP4, WP5-8	M20	M24
B.14	**Pilots' Functional verification**	WP4, WP5-8	M20	M24
B.15	**Integration and testing -2-**	WP4, WP5-8	M20	M24
B.15.1	Testing individual components	WP4, WP5-8	M20	M24
B.15.2	Testing APIs & Interfaces	WP4, WP5-8	M20	M24
B.15.3	Integration of all components	WP4, WP5-8	M20	M24
B.15.4	Integrated platform lab testing	WP4, WP5-8	M20	M24
B.15.5	Integrated platform field testing	WP4, WP5-8	M20	M24

B.15.6	Intermediate TwinEU prototype demo testing	WP4, WP5-8	M20	M24

Table 12: Final Development Phase

C	**Final Development Phase**			M36
C.1	**Cybersecurity and data privacy requirements Final version**	T3.4	M18	M30
C.2	**Finalisation of Functional & Technical Requirements**	WP3, WP4	M18	M26
MS7	**Activities completed in demo WPs**	WP4		M30
C.3	**Coordination with demos and data sharing strategies – Final cycle of coordination**	T4.3	M24	M33
C.4	**Middleware development - Final version**	T4.4	M24	M33
C.5	**EU Data Space adaptation to support the TwinEU continuum – Final release**	T4.5	M24	M33
C.5.1	Testing and validation of the new features of the OneNet Connector to serve the DTs	T4.5	M24	M33
C.5.2	Testing and validation of the interoperability requirements, data integration and homogenization	T4.5	M24	M33
C.5.3	Mapping Data Models into NGSI-LD – Final version	T4.5	M24	M33
C.6	**Final integration of the TwinEU framework, including the Service Workbench, the Middleware and the Connector.**		M24	M33
C.7	**Ensure reference architecture compliance**	WP4	M24	M33
MS8	**Preliminary validation of TwinEU platform**	WP4 – WP8		M30
C.8	**Final assessment**	WP4	M24	M33
C.9	**2nd & final release integration**	WP4	M24	M33
C.10	**Integration and testing -3-**	WP4 – WP8	M24	M33
C.10.1	Testing individual components	WP4 – WP8	M24	M33
C.10.2	Testing APIs & Interfaces	WP4 – WP8	M24	M33
C.10.3	Integration of all components	WP4 – WP8	M24	M33
C.10.4	Integrated platform lab testing	WP4 – WP8	M24	M33
C.10.5	Integrated platform field testing	WP4 – WP8	M24	M33
C.10.6	Final TwinEU prototype testing	WP4 – WP8	M24	M33
C.10.7	TwinEU Validation	WP4 – WP8	M24	M33
C.10.8	TwinEU integrated platform validation	WP4 – WP8	M24	M33
C.10.9	Post-integration testing	WP4 – WP8	M24	M33
C.11	**WP4 Tech Release deployment activities**	WP4	M24	M33
C.12	**Pilot test & validation of final integrated release**	WP4 – WP8		M36
D4.3	**Validation of platform & implementation of the TwinEU pan-European scenarios**	T4.6	M25	M36
MS9	**Availability of TwinEU final exploitation plan**	T4.6, WP10		M36

7.2 Detailed Planning

7.2.1 Phase A: First Development Phase

Definition of Tech Release Components

This stage involves bilateral and dynamic discussion among the TwinEU technical stakeholders to define the technology enablers and components that will support the project's main objectives.

Technical and Functional Requirements definition

This stage will analyse the outcomes of key activities - Task 2.1 (Digitalisation challenges and opportunities in the system planning, operation, and energy markets), Task 2.3 (Digital technologies and boundary conditions and requirements) and the analysis of similar European projects – to develop an initial set of technical requirements of TwinEU. Additionally, a high-level review of Task 2.4 (Setting priorities and development of use cases) and Task 2.5 (Definition of validation process, pan-European scenarios and KPIs) supports the preliminary planning of the functional requirements.

Reference Architecture Design – First version

This stage focuses on defining a foundational structure of the TwinEU software framework. It involves creating a high-level architectural model that outlines core technical and structural components including their relationships, communication interfaces, and deployment environments. The initial reference architecture serves as the baseline for all future versions, all the development activities. It ensures alignment with European Data pace and DT principles. The TwinEU architecture will focus on the TwinEU Middleware, the TwinEU Connector and the TwinEU services Workbench and the relationship between them within a Data Space ecosystem.

Extrapolation of Functional Specifications & review

In this stage the outcomes of the initial functional requirements definition are updated to the organized identification and formalization of the preliminary functional capabilities of the TwinEU system based on stakeholder needs, use-case requirements, and architectural design. This extrapolation ensures traceability between high-level objectives and concrete software functionalities. The initial set of requirements will focus on the following:

- Assessment of demo specific use cases.
- Middleware Functional Requirements definition & review.
- Analysis of requirements for energy data space compliant technological enablers.
- Analysis of requirements for energy services and DTs.
- Analysis of requirements related to the service Workbench / Big Data / AI Marketplace.
- Analysis of requirements related to Multi-user XR development platform.
- Analysis of the functional requirements of the Federated DTs concept.

Technical Requirements Specification

In this stage the outcomes of the initial functional requirements definition are updated to the organized identification and formalization of the technical requirements of the TwinEU system based on the desired functional requirements that are defined. This stage focuses on defining performance constraints, interoperability protocols, data handling mechanisms, scalability targets, and security policies. These requirements serve as guidelines and validation criteria throughout the lifecycle of the project.

Advanced service Workbench / Big Data / AI Marketplace – First Version

This phase focuses on the initial design and development of the services and models that will be used in the TwinEU service Workbench and the Big Data and AI Marketplace. In this stage there will be a thorough analysis of the pilot needs, and these services will be aligned with them. Additionally, there will be research on useful tools and datasets among the TwinEU stakeholders that can support these activities.

Multi-user XR development platform – First Version

This phase focuses on the initial development of the Interactive augmented exploration of TwinEU DTs. In this stage there will be a thorough analysis of the already implemented DTs or the design of the future DTs to conclude in the initial models that will be integrated within Unity3d and the initial design of the Graphical user interface that will be used for the application.

Middleware development - First version

This stage of development includes a first prototype of the TwinEU Middleware that will include core functionalities such as basic identity management, data access control, communication protocols, and peer-to-peer data exchange. This version will be used for the first use case assessment by some test users. The basic development effort will be:

- Base connector implementation.
- Interfaces & APIs development.
- Integration & homogenisation middleware sub layer implementation – first version.
- User interface base version.
- Basic data usage and access policy.
- Basic Identity and authorization tools.

Coordination with demos and data sharing strategies – First cycle of coordination

This stage includes coordination activities that involve the identification of the TwinEU stakeholders that will participate as data or service providers and data or service consumers in the desired use-case scenarios. These stakeholders should be familiar with data space principles and the tools that enable data and service sharing. For this reason, in this stage the technical stakeholders should provide specific demonstrations and presentations to support this goal.

EU Data Space adaptation to support the TwinEU continuum

This stage focuses on the adoption of EU data space approaches focusing on data management, security, sovereignty and trust regarding the TwinEU components. Specifically, this stage should define the ground rules and components that will be used within TwinEU regarding to model sharing, data sharing governance, data integration and homogenization and the implementation of a data connector according to the IDSA principles.

Initial functional validation & testing

In this stage it is performed a basic functional testing to validate the implementation of core features regarding the specified functional and technical requirements. These test scenarios include data transfers, authentication workflows, service registration, and user interactions.

Integration and testing -1-

The goal of this phase is to develop the TwinEU deployment guide, which will be included in the updated version of the Integration Plan following the creation of the initial middleware prototype.

This early prototype will undergo laboratory testing to verify that the TwinEU middleware meets the functional requirements that are defined within the phase A.

7.2.2 Phase B: Intermediate Development Phase

Reference Architecture Design fine-tuning and finalization

This stage focuses on the refinement of the initial architectural version of the TwinEU system and the finalisation of it. This stage may contain updated interface specifications between the components, changes on the functionality and the responsibility of each component and generally improved workflows, technology enablers and use-case scenarios. Overall, these changes should be perfectly aligned with the functional requirements, technical requirements and stakeholder feedback that are defined in other tasks.

Cybersecurity and data privacy requirements first version

In this stage there will be performed the first attempt of forming the ground legal rules, standards and regulations regarding the TwinEU ecosystem. At this first stage there will be an analysis of the European standards especially regarding the data protection that will lead to the initial requirements.

Functional Specifications Definition final version

In this stage, the functional requirements of the TwinEU framework will be further developed by expanding the scope of the initial prototype and conducting a comprehensive analysis of all aspects of TwinEU. The final set of requirements will focus on the following:

- Alignment with the whole demo use cases.
- Middleware Functional Requirements finalisation & review.
- Finalisation of requirements for energy data space compliant technological enablers.
- Finalisation of requirements for energy services and DTs.
- Finalisation of requirements related to the service Workbench / Big Data / AI Marketplace.
- Finalisation of requirements related to Multi-user XR development platform.
- Finalisation of the functional requirements of the Federated DTs concept.

Technical Specifications Definition final version

In this stage the technical requirements that were defined in the Phase A are updated and finalised. These completed technical requirements will focus on updating performance metrics, resulting in the desired interoperability standards, improving data handling and processing and resulting in the best possible federated deployment mechanism.

Implementation of advanced service Workbench / Big Data / AI Marketplace – Final Version

This development iteration will deliver the final version of the Services and models that will be deployed in the TwinEU Workbench. This development will be based on the careful analysis of all the pilot use cases and the feedback from the TwinEU stakeholders about the development results of the Phase A. Additionally, during this stage will be deployed the IDSA Federated Catalogue into the implementation of Services Workbench. Finally, this stage requires the integration of the Workbench with the rest TwinEU components.

Implementation of interactive augmented exploration of TwinEU DT – Final Version

This development iteration will deliver the final version of the models and visualization tools that will be deployed in the TwinEU Multi-user XR Development Platform. This development will be based

on the careful analysis of all the pilot needs and the feedback from the TwinEU stakeholders about the development results of the Phase A. Additionally, during this stage will be implemented the Graphical user Interface and the real-time data integration tools into the implementation of the XR Platform. Finally, this stage requires the integration of the XR Platform with the rest TwinEU components.

Middleware development - Intermediate version

During this stage the basic development activities will include the updates and the improvement of the first Middleware prototype. First, it is crucial for the Middleware to align with the defined functional, technical requirements and the TwinEU architectural design. The basic development effort will be:

- Final connector implementation.
- Interfaces & APIs development.
- User interface updated version.
- New features for trusted data exchange protocols and metadata-based routing.
- Updated data usage and access policy.
- Updated Identity and authorization tools.

Coordination with demos and data sharing strategies – Second cycle of coordination

Until this stage the TwinEU stakeholders should have been already educated to use the data sharing tools and the settings of the components. During this stage the TwinEU stakeholders should begin the integration of their services, data or DTs in the data sharing components and perform the first test data sharing attempts with test data or for more mature integrations with real data. Of course, this stage includes the feedback between the users and the technical teams.

EU Data Space adaptation to support the TwinEU continuum

During this stage the main objective is to connect the DT that is the main entity within TwinEU with the main entity of Data Space ecosystem which is the Connector. Hence, it is essential to perform the integration mechanism of a DT to the TwinEU Connector to enable the exchange of data and models under standard predefined principles of sovereignty, access policy and data interoperability.

Pre-integration testing

During this stage all crucial components and subsystems undergo a standalone integration testing. These tests include interface mismatches identification, error handling and generally the compatibility check of the components.

1st & 2nd Release integration

This stage combines features that are integrated during the initial integration process and features from the second integration release. The objective of this stage is the harmonization of modules and a stable deployment and alignment of the two versions.

Ensure reference architecture compliance

During this stage a formal validation is performed to confirm that all the components and functionalities adhere to the final reference architecture that is defined in previous steps.

Post-integration testing

This stage follows the process of the system integration and includes a comprehensive testing across all system layers to verify stability, functional correctness, and performance.

Pilots Operation Testing

In this stage the TwinEU integrated system is deployed within the TwinEU pilots to evaluate real-world performance, accuracy of data flows, and system usability. The objective of this phase is to identify specific issues and insights through end-user operational integration.

Pilots' Functional verification

This stage validates all the TwinEU system features and workflows against the predefined use cases and functional specifications of the pilots. This stage confirms that the deployed system satisfies policy requirements, end-user expectations, and technical constraints.

Integration and testing -2-

The goal of this concluding phase is to develop the second, updated prototype of the TwinEU platform by taking into consideration the outcomes of the previous stages of Phase B and integrating all individual components into a unified framework. The platform will undergo testing in both field and laboratory environments, with initial evaluations conducted to ensure an overall compliance with Phase B architecture and requirements and to confirm the successful execution of demonstration use cases.

7.2.3 Phase C: Final Development Phase

Cybersecurity and data privacy requirements Final version

This task finalizes the comprehensive sovereignty, cybersecurity, trust, and data privacy framework for the TwinEU system. During this stage it is ensured that all components and data communication comply with the state-of-the-art legal and regulatory rules, including GDPR and EU Cybersecurity obligations.

Finalisation of Functional & Technical Requirements

During this stage all the functional and technical specifications are summarized, finalized and reviewed. This step is also very important because of the comprehensive documentation that is necessary for the correct guidance of rest implementations and deployments and for the long-term maintenance planning.

Coordination with demos and data sharing strategies – Final cycle of coordination

This final cycle of the coordination with the demonstration sites regarding the data sharing, focuses on aligning the final system behaviour with the overall operational workflows and data sharing activities of the pilots. A prerequisite of this step is the pilot stakeholders to be familiar with the system functionality and to have completed successfully the preparation processes that were described in the previous phases.

Middleware development - Final version

During this stage will be developed the final version of the middleware, incorporating updates and improvements based on:

- Issues identified during the second integration and testing cycle, and
- any new changes arising from the finalised functional requirements and use cases.

EU Data Space adaptation to support the TwinEU continuum – Final release

This stage implements the last release of the TwinEU Data Space which integrates the whole service utilization, data and models exchange process through the Data Space Connectors. Additionally, semantic and ontology orchestration should be optimized, aligning with IDSA compatible standards such as SAREF, ETSI Context and NGSI-LD. Special mention should be given to the Adaptive DT concept that should be totally compatible, scalable and extendible through the Data Space Connector.

Final integration of the TwinEU framework, including the Service Workbench, the Middleware and the Connectors.

This task completes the integration of all crucial system components into a single cohesive framework. The main objective is to ensure full interoperability between the Service Workbench, the Connectors and Middleware, enabling seamless data and service exchanges across the TwinEU ecosystem.

Ensure reference architecture compliance

This stage performs a final compliance audit and cross examination that the implemented system architecture conforms to the approved and finalised TwinEU reference architecture.

Final assessment

In this stage a comprehensive evaluation of the TwinEU system is carried out. This evaluation includes technical performance reviews, system maturity analysis, interfaces testing, user acceptance feedback and readiness assessment for operational deployment.

2nd & final release integration

In this phase all the components and demo implementations are fully integrated into the final system release.

Integration and testing -3-

This stage is the final round of system integration and testing. This phase validates overall system performance and verifies that all components function correctly under production-like conditions.

WP4 Tech Release deployment activities

This stage performs the deployment of the final technical release managed under WP4. The activities of this stage include documentation, packaging, deployment support, and communication with stakeholders to facilitate the launching in various operational environments.

Pilot test & validation of final integrated release

After the completion of integration and pre-release test processes, the final integrated release is tested and validated in live pilot environments. This ensures that all end-to-end functionalities are working correctly, data flows are seamless, and the platform delivers the expected value to end users under realistic operational scenarios.

8 Conclusions

Concluding, the work presented in this deliverable is the first iteration of three several consecutive development processes (and the deliverables that present their steps and methodologies) that aim at designing and iteratively developing the evolution on the TwinEU Platform implementation and deployment, alongside with its role in the broader TwinEU framework. The comprehensive analysis provided in this report of the TwinEU project, demonstrates a strong effort of all the project partners towards technological advancement and alignment towards a robust Data-Space-enabled federated ecosystem enriched with value-added functionalities.

The first version of the TwinEU Platform is described, along with its initial alignment, interactions and integration with the value-added components of the TwinEU framework, such as the Services Workbench/ Big Data / AI marketplace, the Interactive tools for augmented exploration of TwinEU DT and the Data Space components (basically the TwinEU Connector). This is important information for the TwinEU participants as an introduction for the deployment, integration and facilitation needs of the TwinEU Platform and related components. This first effort is the basis that will ensure the enhancement and extension of the TwinEU framework that will be implemented in an iterative design and development process until the end of the project.

Based on the results of this first version, it is recommended that future work continues to monitor and emphasize to modularity, scalability, and compliance with data and model management and sovereignty principles. Additionally, there is a need for a constant monitoring to and enrichment with the evolving functionalities of the TwinEU Data Space Connectors, in order to cover all the TwinEU use cases and offer sustained interoperability and stakeholder trust. Moreover, continued collaboration with demonstration activities and the service layer (AI marketplace, Big Data workbench) will further strengthen the utility and reach the final desired version of the TwinEU platform.

Looking forward, the first development outcomes of the TwinEU platform create confidence in the creation of a pan-European Data Space ecosystem with a variety of cutting-edge, value-added features. The TwinEU whole Framework, which is totally dependent on the TwinEU platform, is expected to evolve as a cornerstone for Europe’s digital energy transition. It has the potential to become a reference framework not only for technical integration but also for governance and policy alignment, enabling an energy system that is resilient, intelligent, and cooperative by design.

References

[1] Gaia-X, “Home - Gaia-X: A federated Secure data infrastructure”, https://gaia-x.eu/.

[2] FIWARE, “FIWARE - Open APIs for Open Minds,” https://www.fiware.org/.

[3] BRIDGE, “Smart Grid Storage Systems Digital Projects”, https://bridge-smart-grid-storage-systems-digital-projects.ec.europa.eu/.

[4] International Data Spaces Association, “Home - International Data Spaces”, https://internationaldataspaces.org/

[5] OneNet Project: https://www.onenet-project.eu/.

[6] BD4NRG Project, https://www.bd4nrg.eu/.

[7] Enershare Project, https://enershare.eu/.

[8] ISO, “ISO/IEC/IEEE 42010:2022 -Software, systems and enterprise — Architecture description,” [Online]. Available: https://www.iso.org/standard/74393.html.

[9] P. Kruchten, “Architectural Blueprints—The “4+1” View Model of Software Architecture,” IEEE Software 12 (6), pp. 42-50, 1995, doi: 10.1109/52.469759.

[10] European Commission, “European Energy Data Exchange Reference Architecture,” BRIDGE - Data Management Working Group, 2020. [Online]. Available: https://energy.ec.europa.eu/system/files/2021-06/bridge_wg_data_management_eu_reference_architcture_report_2020-2021_0.pdf.

[11] CEN – CENELEC – ETSI: Smart Grid Coordination Group, “Smart Grid Reference Architecture Report V2.0”, 2012, https://www.researchgate.net/publication/263264218_CEN_-_CENELEC_-_ETSI_Smart_Grid_Coordination_Group_-_Smart_Grid_Reference_Architecture_Report_20.

[12] Comprehensive Architecture for Smart Grid (COSMAG), “Definition Of A Global Architecture For Smart Grid Applications”, 2018, https://aioti.eu/wp-content/uploads/2019/03/20181010_COSMAG_07.pdf.

[13] FIWARE, “FIWARE: The Open Source Platform Of Choice For Building Smart Energy Solutions”. [Online]. Available: https://www.fiware.org/wp-content/directories/marketing-toolbox/material/FIWAREBrochure_SmartEnergy.pdf.

[14] Alliance for Internet of Things Innovation “High Level Architecture (HLA), Release 5.0”, December 2020. [Online] Available: https://aioti.eu/wp-content/uploads/2020/12/AIOTI_HLA_R5_201221_Published.pdf.

[15] Big Data Value Association, “Towards A European Data Sharing Space: Enabling data exchange and unlocking AI potential, BDVA Position Paper”, April 2019. [Online]. Available: https://bdva.eu/wp-content/uploads/pdfs-legacy/BDVA%20DataSharingSpace%20PositionPaper_April2019_V1.pdf.

[16] GAIA-X, “Gaia-x - Architecture Document – 22.04 Release,” Apr. 2022. [Online]. Available: https://gaia-x.eu/wp-content/uploads/2022/06/Gaia-x-Architecture-Document-22.04-Release.pdf

[17] MinIO, Inc., “S3 Compatible Storage for AI,” https://min.io/.

[18] TwinEU, “D3.1: TwinEU Open Reference Architecture”, 2025, https://twineu.net/deliverables/.

[19] RabbitMQ, https://www.rabbitmq.com/.

[20] Apache NiFi. https://nifi.apache.org/.

[21] Unity Real-Time Development Platform, https://unity.com/.

[22] Unreal Engine, https://www.unrealengine.com/en-US.

[23] WebXR Device API Specification, https://github.com/immersive-web/webxr.

[24]OpenXR, https://www.khronos.org/OpenXR.
[25]Meta Business SDK, https://developers.facebook.com/docs/business-sdk/.
[26]Khronos gITF, https://www.khronos.org/Gltf.
[27]Unity, "About Netcode for GameObjects" https://docs-multiplayer.unity3d.com/netcode/current/about/.
[28]Data Space Support Centre, "Data, Services, and Offerings Descriptions," 2024. [Online]. Available: https://dssc.eu/space/bv15e/766069419/Data,+Services,+and+Offerings+Descriptions.
[29]International Data Space Association, "Contract Negotiation | Specification", https://docs.internationaldataspaces.org/ids-knowledgebase/dataspace-protocol/contract-negotiation/contract.negotiation.protocol.
[30]Code with C, "C++ and Virtual Reality: Developing Real-Time VR Systems", https://www.codewithc.com/c-and-virtual-reality-developing-real-time-vr-systems/.
[31]Powervr-Graphics, "GitHub - powervr-graphics/Native_SDK: C++ cross-platform 3D graphics SDK.", https://github.com/powervr-graphics/Native_SDK.
[32]TwinEU "D2.2: TwinEU Use Cases, pan-European scenarios and KPIs", 2025. [Online] Available: https://twineu.net/wp-content/uploads/2025/07/TwinEU_D2.2_v1.0-3.pdf.