Funded by
the European Union

HORIZON-CL5-2022-D4-02

EUROPEAN COMMISSION

European Climate, Infrastructure and Environment Executive Agency

Grant agreement no. 101123238

Smart Grid-Efficient Interactive Buildings

Deliverable D3.7

Integrated EVELIXIA Middleware Layer's and Orchestration

Project acronym	EVELIXIA
Full title	Smart Grid-Efficient Interactive Buildings
Grant agreement number	101123238
Topic identifier	HORIZON-CL5-2022-D4-02-04
Call	HORIZON-CL5-2022-D4-02
Funding scheme	HORIZON Innovation Actions
Project duration	48 months (1 October 2023 – 30 September 2027)
Coordinator	ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS (CERTH)
Consortium partners	CERTH, RINA-C, CEA, CIRCE, UBE, HAEE, IESRD, UNIGE, SOLVUS, R2M, EI-JKU, FHB, EEE, EG, ÖE, PINK, TUCN, DEER, TN, ENTECH, SDEF, EGC, KB, AF, Sustain, NEOGRID, MPODOSAKEIO, DHCP, HEDNO, BER, MEISA, ITG, NTTDATA, TUAS, NEOY, HES-SO
Website	https://www.evelixia-project.eu/
Cordis	https://cordis.europa.eu/project/id/101123238

VELIXIA
BUILDINGS AS ACTIVE UTILITY NODES

Funded by the European Union

Disclaimer

Copyright Message

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon Europe Framework Programme for Research and Innovation under grant agreement no 101123238. ***Disclaimer:*** *The European Commission is not responsible for any use made of the information contained herein. The content does not necessarily reflect the opinion of the European Commission.*

VELIXIA
BUILDINGS AS ACTIVE UTILITY NODES

Funded by the European Union

Deliverable D3.7

Integrated EVELIXIA Middleware Layer's and Orchestration

Deliverable number	D3.7
Deliverable name	EVELIXIA platform external communication and common information management
Lead beneficiary	CIRCE
Description	This deliverable is directly linked to the activities foreseen in Task 3.6, consolidating all foreseen technical developments on EVELIXIA's middleware layer brokerage, orchestration, workflow management and components functional integration. This report is considered as the first version of D3.8
WP	WP3
Related task(s)	T3.6
Type	Report
Dissemination level	Public
Delivery date	March 2025 (M18)
Main authors	Alberto Moreno (CIRCE)
Contributors	

Document history

Version	Date	Changes	Author
V1 – first draft shared with contributing partners and reviewers	28/01/25		CIRCE
V1 – reviews	February '25		
V1 – consolidated version	February '25		CIRCE
2nd review	February '25		NEOGRID, SOLVUS, CERTH, ENTECH, CEA
V2 – second draft	February '25		CIRCE
Final version	February '25		CIRCE
Final deliverable submission	M18 (March 25)		CIRCE

ABBREVIATIONS

Abbreviation	Name
API	(Application Programming Interface): A set of protocols and tools for building software and applications, allowing different systems to communicate with each other.
CIM	(Common Information Model): A standard for representing data and information in a consistent and interoperable manner, often using ontologies like SAREF.
NLP	(Natural Language Processing): A field of artificial intelligence that focuses on the interaction between computers and humans through natural language.
Pub/Sub	(Publish/Subscribe): A messaging pattern where senders (publishers) send messages to a topic, and receivers (subscribers) receive messages from that topic.
SAREF	(Smart Appliances REFerence ontology): An ontology developed to enable interoperability between smart appliances and other devices.
SPARQL	(Protocol and RDF Query Language): A query language used to retrieve and manipulate data stored in Resource Description Framework (RDF) format.

GLOSSARY OF TERMS

Abbreviation	Name
Authentication	The process of verifying the identity of a user or system.
Authorization	The process of granting or denying access to resources based on the authenticated identity.
Data Broker	A middleware component that manages the distribution and routing of data between different parts of the system, often using pub/sub mechanisms.
Deployment Pipeline	A series of automated processes that manage the deployment of software from development to production environments.
Field Layer	The layer in an architecture that includes devices and sensors collecting data from the physical environment.
Knowledge Graph:	A structured representation of data that enables semantic queries and insights, often using technologies like SPARQL.
Northbound Open API Connector	A module that facilitates communication between the data management layer and the application/services layer.
Ontology	A formal representation of knowledge as a set of concepts and the relationships between those concepts.
Security Measures	Techniques and practices implemented to protect data and systems from unauthorized access and threats.
Semantic Query	A query that uses semantic technologies to retrieve data based on the meaning and relationships of the data.
Service Broker	A component that manages service requests and responses between different parts of the system.
Southbound Open API Connector	A module that facilitates communication between the data management layer and field layer devices.

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This document outlines the testing and integration processes for the data management layer, focusing on the verification of functionalities and the seamless interaction between software components which have been developed to efficiently collect, process, and expose data from pilot sites.

The testing and integration process begins with unit testing to verify the functionality of individual components, including communication protocols, data aggregation logic, and security measures, ensuring they operate correctly in isolation, as well as verifying data quality, security and consistency.

This is followed by testing the interactions between integrated components, validating data flow between pilot sites APIs and the workflows for data ingestion, and routing, under various load conditions through load and stress testing to ensure it can handle the expected volume of data and user interactions.

By adhering to these testing and integration processes, the data management layer is validated to meet the requirements of collecting, processing, and exposing pilot sites data, ensuring data integrity, security, and accessibility through the EVELIXIA's architecture.

1 INTRODUCTION AND OBJECTIVES

This chapter defines the objectives, scope, context, and structure of the deliverable. It also outlines its relationship with other tasks within the project. To this end D3.7 delivers the Integrated EVELIXIA Middleware Layer's and Orchestration, according to the Integration of EVELIXIA's Middleware layer components (T3.6).

1.1 Scope and Context

This deliverable is directly linked to the activities foreseen in Task 3.6, consolidating all foreseen technical developments on EVELIXIA's middleware layer brokerage, orchestration, workflow management and components functional integration.

The scope of this document, titled " Integrated EVELIXIA Middleware Layer's and Orchestration," encompasses the integration of the technical developments and deployments within the data management layer of the EVELIXIA platform described in D3.1 titled "EVELIXIA platform External Communication and Common Information Management".

Moreover, it aims to provide a comprehensive overview of the integration of all the developed software components, by detailing the test scenarios, of each software component, this document provides a clear framework for understanding how the data management layer supports the overall EVELIXIA platform architecture.

The integration document focuses on how the different components interact and work together as a cohesive system within the pilot sites data management framework:

- Integration strategies and end-to-end workflows.
- Detailed explanation of data flow between components.
- Comprehensive testing strategy to ensure seamless interaction and functionality of the entire system.

1.2 Interaction with other Tasks and Work Packages

Deliverable D3.7 concisely presents the integration of the technical developments within the data management layer, based on input from:

- ✓ WP1 (D1.3 & D1.7)

The software components developed in tasks T3.1 and T3.2 are designed to interact closely with and depend on the developments from:

- ✓ WP2 (T2.4)
- ✓ WP3 (T3.3, T3.4&T3.5)
- ✓ WP4 (T4.6)
- ✓ WP5 (T5.2)

1.3 Structure of this deliverable

The deliverable is constituted by the following chapters which are interrelated and provide an overall analysis of the EVELIXIA platform architecture.

- Chapter Introduction and Objectives
- Chapter Integration Strategy: environment setup, and test scenarios
- Chapter Software Components Integration: Details on key components interaction like the Southbound and Northbound Open API Connectors, Data Broker, Knowledge Graph, and Common Information Model (CIM).
- Chapter Integration with external components: data broker and service broker end-to-end.
- Chapter Test Scenarios: incremental approach to consolidate each software component after being deployed
- Chapter Conclusion: Summary and future enhancements.
- Chapter References

2 INTEGRATION STRATEGY

By following this approach, it has been ensured that each development works correctly on its own and that the entire system operates seamlessly when all components are integrated:

1. **Development Phase**: Each software component is developed individually. During this phase, the focus is on ensuring that each component functions correctly on its own.

2. **Deployment and Testing**: Each component is deployed and tested multiple times. This iterative process continues until the component's functionality meets the expected criteria. Any issues identified during testing are resolved, and the component is refined accordingly.

3. **Integration Testing**: Once all components have been individually tested and validated, the next step is to test them together. This involves deploying all components in a shared environment and conducting tests to observe how they interact with each other. The goal is to ensure that the integrated system functions as expected and that there are no issues arising from the interactions between components.

The first two phases are fully described in **D3.1** “EVELIXIA platform external communication and common information management”.

2.1 Objective

To ensure seamless interaction and data flow between the Southbound Open API Connector, Data Broker (Apache Airflow), Northbound Open API Connector, and Knowledge Graph (GraphDB with SAREF ontology).

2.2 Approach

- **Incremental Integration**: Integrate components incrementally, starting with the Southbound Open API Connector and Data Broker, followed by the Knowledge Graph and finally the Northbound Open API Connector

- **Continuous Testing**: Implement continuous testing at each integration stage to identify and resolve issues early.

2.3 Steps

1. **Initial Setup**:
    - Deploy each component in isolated environments using Docker.
    - Ensure that each component is independently functional and passes its respective tests.
2. **Component Integration**:
    - Integrate components in a step-by-step manner, verifying data flow and functionality at each stage.
3. **End-to-End Integration**:
    - Perform end-to-end tests to verify that data flows seamlessly from pilot sites data to the service broker.
    - Ensure that all components interact correctly, and that data is accurately collected, processed.
4. **Data flow**
    - The Southbound Open API Connector will access pilot site to gather RAW data (POST -raw JSON data-) RAW ENDPOINT 1. This has been achieved for M18.
    - T3.3 transformation modules will read the RAW data through GET request to the Southbound Open API RAW ENDPOINT 1. This preliminary interaction has been achieved for M18.
    - T3.3 transformation modules will store healed data through POST requests which will store healed data in dedicated PostgreSQL database. This will try to be also achieved for M18.
    - Data Broker will insert healed data into the knowledge graph database (GraphDB) through a POST to the HEALED ENDPOINT 2-
    - Service broker API will read healed data through GET HEALED ENDPOINT 2

3 SOFTWARE COMPONENTS INTEGRATION

This section provides a detailed overview of the key components developed within the data management layer already described in D1.3. Each component plays a crucial role in facilitating seamless data integration and communication across the EVELIXIA platform. The developments include the Southbound and Northbound Open API Connectors, which enable interaction with field devices and application/services layers, respectively.

Additionally, the Data Broker manages data routing and pub/sub mechanisms, while the Knowledge Graph offers semantic query capabilities for deeper insights. The Common Information Model (CIM) standardizes data exchange, ensuring interoperability and consistency.

These components form a robust middleware layer that enhances the EVELIXIA platform's overall functionality and scalability.

The integration of these components creates a cohesive system that efficiently manages Pilot Sites data from collection to exposure. Comprehensive testing, including unit, integration, and performance testing, ensures the system meets the required standards for functionality, reliability, and scalability.

3.1 Southbound Open API Connector and Data Broker

- **Integration Steps**:
  - Configure the Data Broker (Apache Airflow) to periodically fetch data from the Southbound Open API Connector.
  - Verify data flow by checking that the Data Broker successfully ingests and processes data from the connector.
- **Testing**:
  - Perform data collection and ingestion tests to ensure accurate data capture and processing.
  - Validate the scheduling and execution of workflows in Apache Airflow.

3.2 Data Broker and Northbound Open API Connector

- **Integration Steps**:
  - Configure the Northbound Open API Connector to fetch data from the Data Broker.
  - Verify data flow by checking that the Northbound Open API Connector successfully retrieves, transforms, and exposes data.
- **Testing**:
  - Perform data retrieval, transformation, and exposure tests to ensure data integrity and accuracy.
  - Validate the security mechanisms for data access and transmission.

3.3 Northbound Open API Connector and Knowledge Graph

- **Integration Steps**:
  - Configure the Knowledge Graph component to ingest data exposed by the Northbound Open API Connector.
  - Verify data flow by checking that RDF data is correctly ingested into GraphDB and that SPARQL queries return expected results.
- **Testing**:
  - Perform data ingestion and SPARQL query tests to ensure correct data integration and querying.
  - Validate the performance and scalability of the knowledge graph.

4 INTEGRATION WITH EXTERNAL COMPONENTS

4.1 Interaction with field layer devices

For this first interaction (M18), these have been the selected pilot sites: Greek, Austrian and Danish. This decision is based on the availability and matureness of both data and metadata, apart from being well documented and ready to use.

Objective:

- To integrate the developed API (T2.4) of various pilot sites to obtain RAW data from field devices.

Integration Steps:

1. **API Configuration**:
   - Identify and configure the APIs provided by different pilot sites for accessing raw data from Pilot sites data.
   - Ensure that the APIs support the required communication protocols (HTTPS) and data formats (JSON, JSON-LD).
   - Implement comprehensive API documentation using tools like Swagger to facilitate understanding and integration for developers. Ensure the documentation includes detailed information on endpoints, parameters, expected responses, and error codes to enhance usability and reduce integration time.

2. **Data Collection**:
   - Implement the Southbound Open API Connector to periodically fetch raw data from the configured APIs.
   - Use task scheduling (Apache Airflow) to automate the data collection process at specified intervals.

3. **Data Aggregation**:
   - Aggregate data from multiple field devices to provide a unified view.
   - Ensure data consistency and accuracy during the aggregation process.

4. **Error Handling**:
   - Implement error handling mechanisms to manage communication failures or data inconsistencies.
   - Log errors and retry data collection as necessary.

Testing:

- **Data Collection Test**: Verify that the Southbound Open API Connector can successfully fetch data from the APIs of the selected pilot sites.

- **Data Aggregation Test**: Ensure that data from multiple field devices is correctly aggregated.
- **Error Handling Test**: Simulate API failures and verify that the connector handles errors gracefully.

4.2 Interaction with Application/Services layer

Objective:

- To facilitate the interaction between the Data Broker (middleware data management layer) and the service broker through the Northbound Open API Connector, using a publish-subscribe mechanism.

Integration Steps:

1. **Data Publishing**:
    - Configure the Northbound Open API Connector to expose healed data to the service broker.
    - Implement data transformation and healing rules to ensure data quality before publishing.

2. **Service Broker Subscription**:
    - Set up the service broker to subscribe to the data published by the Northbound Open API Connector.
    - Future use in the second prototype (M33) of a publish-subscribe mechanism (MQTT to Kafka topics) to manage data flow between the Data Broker and the service broker.

3. **Data Routing**:
    - Ensure that the Data Broker routes transformed data to the Northbound Open API Connector for publishing.
    - Implement dynamic routing based on predefined rules and conditions.

4. **Security and Access Control**:

- Implement authentication and authorization mechanisms (T4.5) to secure data access.
- Ensure data integrity and confidentiality during transmission through SSL certificates.

Testing:

- **Data Publishing Test**: Verify that the Northbound Open API Connector can successfully publish healed data to the service broker.
- **Subscription Test**: Ensure that the service broker can subscribe to and retrieve data from the Northbound Open API Connector.
- **Data Routing Test**: Verify that the Data Broker correctly routes data to the Northbound Open API Connector.
- **Security Test**: Ensure that authentication and authorization mechanisms are correctly implemented.

4.3 Interaction with Blockchain APIs

Objective:

- To integrate with blockchain APIs for smart contracts and user management (login/register/tokenize access to resources).

Integration Steps:

1. **Smart Contracts API**:
   - Configure the API to interact with smart contracts on the blockchain.
   - Implement functions to deploy, execute, and query smart contracts.
   - Ensure that data integrity and transaction security are maintained.

2. **User Management API**:
   - Configure the API to handle user login, registration, and tokenization of access to resources.
   - Implement functions for user authentication, authorization, and token management.
   - Ensure secure handling of user credentials and tokens.

3. **Data Integration**:
    - Integrate blockchain data with the existing data management layer.
    - Ensure that data from smart contracts and user management is accurately captured and processed.

4. **Security and Compliance**:
    - Implement robust security measures to protect blockchain transactions and user data.
    - Ensure compliance with relevant regulations and standards.

Testing:

- **Smart Contracts Test**: Verify that the API can successfully deploy, execute, and query smart contracts.
- **User Management Test**: Ensure that the API can handle user login, registration, and tokenization securely.
- **Data Integration Test**: Verify that blockchain data is accurately integrated with the data management layer.
- **Security Test**: Ensure that all security measures are correctly implemented and effective.

5 TEST SCENARIOS

Testing data management layer involves several challenges and complexities due to the nature of the pilot site data, software components and their interactions. Below are the main test scenarios, along with the key challenges and strategies to address them.

By focusing on the following scenarios, a comprehensive overview is provided of how the individual components are combined into a functioning system, ensuring that all interactions and dependencies are properly managed and validated.

5.1 Unit tests

Objective: To verify the functionality of individual components.

Components: Southbound Open API Connector, Data Broker, Northbound Open API Connector, Knowledge Graph.

Tests:

- Data collection, ingestion, transformation, exposure, and querying.
- Use testing frameworks like PyTest (Python) and Mocha (Node.js) for automated unit testing.

Challenges:

- **Data Availability**: Pilot sites data may have intermittent data availability, making it difficult to consistently test data collection.
- **Mocking External Dependencies**: Simulating external APIs and data sources accurately for unit tests.

Strategies:

- Own-generated data was prepared while pilot sites API development was ongoing to simulate sensor data and external dependencies
- Implement retry mechanisms and error handling to manage intermittent data availability.

5.2 Integration tests

Objective: To verify the interaction between integrated components.

Components: Southbound Open API Connector with Data Broker, Data Broker with Northbound Open API Connector, Northbound Open API Connector with Knowledge Graph.

Tests:

- Data flow, transformation, and routing between components.
- Use integration testing tools like Postman for API testing and Airflow's built-in testing capabilities.

Challenges:

- **Data Consistency**: Ensuring data consistency and integrity across multiple components.
- **Complex Workflows**: Managing complex workflows and dependencies between components.

Strategies:

- Implement comprehensive data validation checks at each integration point.
- Use automated tests to validate integration points and ensure data consistency.

5.3 Semantic Query Execution

Objective: To verify the complete data flow from pilot sites data to the service broker.

Components: All components integrated together.

Tests:

- Data collection from sensors, processing through the Data Broker, exposure by the Northbound Open API Connector, and querying in the Knowledge Graph.
- Possibility to use end-to-end capabilities of OpenAI API to convert natural language queries to semantic queries for the second protype (M33).

Challenges:

- **End-to-End Data Flow**: Establishing a complete end-to-end data flow to test the entire cycle of capturing raw data, transforming it, and storing it as healed data.
- **Data Transformation and Ontology Conversion**: Converting data to the SAREF ontology (JSON-LD) and storing it in the knowledge graph (GraphDB).

Strategies:

- Set up a controlled test environment with simulated pilot sites data and data sources (own-generated dummy data in the early stages of the project).
- Use automated scripts to simulate the entire data flow and validate each step.

- Implement detailed logging and monitoring to track data flow and identify issues.

5.4 Performance tests

Objective: To ensure the system can handle the expected load and data volume.

Components: All components.

Tests:

- Load testing, stress testing, and scalability testing.
- Use performance testing tools like JMeter to simulate high load and measure system performance.

Challenges:

- **Scalability**: Ensuring the system can scale to handle large volumes of data and high concurrency.
- **Resource Management**: Managing resource allocation and performance under load.

Strategies:

- Implement horizontal scaling and load balancing to manage high concurrency.
- Use performance monitoring tools to track resource usage and optimize performance.

5.5 Security tests

Objective: To ensure data security and integrity.

Components: Northbound Open API Connector, Data Broker, Knowledge Graph.

Tests:

- Authentication, authorization, data encryption, and vulnerability scanning.

Challenges:

- **Data Protection**: Ensuring data is protected during transmission and storage.
- **Access Control**: Implementing robust access control mechanisms to prevent unauthorized access.

Strategies:

- Use encryption for data at rest and in transit.
- Implement role-based access control (RBAC) and regular security audits.

Apart from securing communications with SSL, RAW data will be read unencrypted, as already stated by the API PS developers.

Basic resources access control through tokenization mechanism.

Robust security measures will be implemented in T4.5 developments to protect data integrity and confidentiality.

Additionally, AES Data privacy will be implemented for the second protype (M33).

6 CONCLUSIONS

The integration of the pilot sites data within the data management layer is a complex yet essential process that ensures seamless interaction and data flow between various components. This document has outlined a comprehensive strategy for integrating and testing each component, addressing these key challenges:

1. **Southbound Open API Connector**:
    - Responsible for collecting RAW data from pilot sites and ensuring timely and reliable data capture.
    - Integrates with the data broker to provide a unified view of sensor data.
2. **Data Broker**:
    - Orchestrates workflows for periodic data ingestion, transformation, and routing through Apache Airflow.
    - Ensures data quality and consistency through comprehensive data validation and error handling mechanisms.
3. **Northbound Open API Connector**:
    - Exposes healed data (transformed in T3.3 modules) to the service broker, ensuring data is clean and validated.
4. **Knowledge Graph and Common Information Model (CIM)**:
    - Utilizes GraphDB to store and manage the SAREF ontology, enabling advanced semantic querying.
    - Provides context insights and relationships from pilot sites data.
5. **Blockchain APIs**:

- Integrates with identity management and NFT to enhance data security and access control, defined in T3.4 & T3.5.
- Ensures secure handling of blockchain transactions and user credentials.

7 REFERENCES

[1] Application Integration: Complete Guide - Gartner: This guide covers application-centric integration, including key technologies, best practices, and the differences between application-centric, data-centric, and event-centric integration approaches Application Integration: Complete Guide | Gartner

[2] Software Integration: Its Importance and Implementation - Adeptia: This comprehensive guide explores the definitions, main components, types, and processes of software integration, along with associated challenges and solutions
Software Integration: Its Importance and Implementation | Adeptia

[3] Software Integration: Benefits, Examples, and Best Practices - NIX United: This article discusses the processes, types, benefits, and examples of software integration, and covers best practices for efficiently integrating applications
https://nix-united.com/blog/guide-to-software-integration-with-examples-types-and-benefits/