D3.4 - Secure Federated Data Management II

Title	D3.4 - Secure Federated Data Management II
Revision Number	1.0
Task reference	T3.1 T3.4
Lead Beneficiary	NUIG
Responsible	
Partners	IQB, NRS, UBI
Deliverable Type	DEM
Dissemination Level	PU
Due Date	2024-12-31 [Month 24]
Delivered Date	2025-01-20
Internal Reviewers	IBM GFT
Quality Assurance	UPRC
Acceptance	Coordinator Accepted
Project Title	FAME - Federated decentralized trusted dAta Marketplace for Embedded finance
Grant Agreement No.	101092639
EC Project Officer	Stefano Bertolo
Programme	HORIZON-CL4-2022-DATA-01-04

This project has received funding from the European Union's Horizon research and innovation programme under Grant Agreement no 101092639

Revision History

Version	Date	Partners	Description
0.1	2024-12-02	ENG	Table of Contents
0.2	2024-12-19	IQB, NRS, UBI	Integrated version with contributions
0.3	2025-01-10	IMB, GFT	Version for peer review
0.4	2025-01-16	NUIG	Update after peer review
1.0	2025-01-20	NUIG	Version for submission

Definitions

Acronyms	Definition
AAI	authentication authorization infrastructure
ACM	Association for Computing Machinery
AI	Artificial Intelligence
API	Application Programming Interface
APM	Assets Policy Management
CD	Continuous Development
CI	Continuous Integration
DLT	Distributed Ledger Technologies
EC	European Commission
EU	European Union
FAIR	Findable Accessible Interoperable Reusable
FAME	Federated decentralized trusted dAta Marketplace for Embedded finance
FDAC	Federated Data Assets Catalogue
GDPR	General Data Protection Regulation
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
IBM	International Business Machines
ID	Identity
IEEE	Institute (of) Electrical (and) Electronic Engineers
JSON	JavaScript Object Notation
JWKS	JSON Web Key Sets
JWT	JSON Web Token
KPI	Key Performance Indicator
ML	Machine Learning
OIDC	OpenID Connect
OS	Operating System
PIN	Personal Identification Number
REST	Representational State Transfer
RSA	Rivest, Shamir, Adelman Cryptographic Algorithm
SA	Supervisory Authority
SDK	Software Development Kit
SME	Small Medium Enterprise
SSI	Server Side Includes
TR	Technical Requirement

UI	User Interface
URL	Uniform Resource Locator
WWW	World Wide Web | W3C = WWW Consortium

Executive Summary

The current digital landscape motivates that software systems include the latest advances on cyber-security and data protection technology, and thus provide better conditions and together with the regulations in relation to data protection, include management operations that ensure the best conditions to ensure systems reliability. In the other hand Federated systems bring new challenges in relation to data protection and data systems control. Federation also demands that user identity and other sensitive information shall be compartmentalized across multiple databases belonging to various organizations, fragmenting the information and exposing personal data imposes elevated risk of cyberattacks and security breaches. This fragmented infrastructure necessitates the frequent creation of new accounts for different platforms and services, such as online marketplaces, leading to a proliferation of potentially outdated and unsecured personal data repositories.

This document outlines the comprehensive work conducted in the FAME project to design, develop, and implement a robust solution to secure the FAME platform, using federated authentication and authorization, along with an asset policy manager for Secured Federated Data Management, both software solutions are necessary to support a Federated infrastructure in the FAME project.

The implemented federated authentication and authorization infrastructure provides a strategic solution for securing FAME applications, simplifying access to multiple services with a single set of credentials. Additionally, the Assets Policy Manager (APM) in systems like FAME plays a key role by integrating with federated identity management, ensuring the efficient and secure management of digital assets while enhancing the user experience.

This deliverable reports the specifications and prototype implementations of the Secure Federated Management Framework. The FAME Secure Federated Data Management Framework comprises the FAME Authentication and Authorisation Infrastructure (FAAID) and the FAME Assets Policy Manager (FAPM) System. The following identified functionalities are highlighted and demonstrated as part of the objectives for the system prototyping reported in this document:

- Streamline user authentication process across different systems and services while maintaining high security and privacy standards.
- Improve user experience, mitigate security risks, and enhance efficiency in controlling access.
- Offer a streamlined, secure, and user-friendly solution for managing digital identities and access controls.
- Enable lifecycle management of policies associated with the federated assets involved in a federated system.
- Facilitate discoverable functionalities through implemented policy management tools, determining what is accessible to be discovered and who is eligible to view and potentially acquire specific assets.
- Provide end-users with a comprehensive list of the assets that they have access to their contents with a clear and consolidated view of their assets, enhancing their ability to manage and track their assets portfolio effectively.

Final note: This deliverable is a continuation of the D3.1 (the first version of Secure Federated Data Management framework), this document aims to be a self-contain specification and some text may be included/repeated as it builds upon the previous deliverable version. However, the specific changes included in this version of the document are detailed in Section 1.4 of this deliverable.

Table of Contents

List of Figures

List of Tables

1 Introduction

In the current digital landscape, user identity information is compartmentalized across multiple centralized databases belonging to various organizations, exposing personal data to an elevated risk of cyberattacks and breaches. This fragmented infrastructure necessitates the frequent creation of new accounts for different platforms and services, such as online marketplaces, leading to a proliferation of potentially outdated and unsecured personal data repositories.

The paradigm of Distributed or Self-Sovereign Identities (SSI) [W3C-DIDs], underpinned by decentralized architectures, presents a transformative resolution to these pervasive challenges. By leveraging blockchain technology and cryptographic principles, SSI places the control of identity data back into the hands of individuals, enabling them to share only what is necessary, with whom they choose, and only for the duration required.

This decentralized identity model significantly mitigates the security risks associated with centralized data storage by eliminating single points of failure. Moreover, it streamlines the user experience by reducing redundant account creation processes, thus simplifying identity verification procedures across a multitude of platforms. Interoperability is inherently designed into these systems, ensuring that identities and credentials can be seamlessly utilized across disparate services and geographic boundaries.

Advancements in this domain are continuously evolving, with research and development focused on enhancing the scalability, user-friendliness, and integration capabilities of SSI solutions. The adoption of SSI represents a significant step towards a more secure, efficient, and user-centric digital identity management framework, realigning the concept of digital identity [EU-DID] with the foundational values of privacy, security, and user control.

On the other hand, where access control is necessary, a suitable solution used is Policy Enforcement Points (PEPs). These are versatile in their application for integration into APIs, microservices and frontend layers, and any other areas within the applications. The widespread incorporation of PEPs throughout the application guarantees frequent and independent verification of authorization at numerous critical points for assets verification and protection. In general Assets Policy Managers (APMs) ensure that other components always display the appropriate assets for authenticated and authorized individuals or organizations. APMs act as a central authority, facilitating the enforcement of access controls throughout the system. This synergistic relationship between federated identity management and the APM asset management capabilities significantly reduces the complexity associated with managing multiple digital identities and assets, offering a more seamless and secure digital experience for both individual users and organizations.

The Assets Policy Manager (APM) functionality, which includes the complete lifecycle management of policies for federated assets, aligns with the objectives of federated systems by simplifying how users and organizations manage access to these assets. The APM enhances the security protocols of federated authentication, ensuring that only authorized individuals can access specific assets, thereby upholding both security and compliance standards.

This document describes the specifications and the implementation of the distributed Identity and Access Management system for a FAME federation of Embedded Finance (EmFi) data marketplaces called Federated Authentication and Authorisation Infrastructure Deployment (FAAID), providing authentication among all the marketplaces and authorization to access FAME resources and to the Federated Assets Policy Manager (FAPM).

1.1 Objective of the Deliverable

The purpose of this deliverable is to present the final version of the FAME Authentication and Authorization Infrastructure (AAI) and the FAME Assets Policy Manager (APM), detailing their implementation and deployment as part of the FAME Marketplace Architecture prototype.

The FAME Authentication and Authorization Infrastructure system's objective is to provide a user-centric authentication based on Self-Sovereign Identity (SSI) and aligning with cutting-edge standardization initiatives such as W3C Decentralised Identifiers (DIDs) v1.0, (the specification is available [1]) [W3C-DIDs] and the ongoing work reported in the W3C Verifiable Claims Working Group (available [2]) [WWW-VCWG]. The FAME AAI approach seeks not only innovation but also widespread commercial acceptance. In striving for comprehensive compliance, we uphold compatibility with widely adopted standards like OpenID Connect (OIDC) [Abie 2022] while staying abreast of the latest advancements.

The objective of the Assets Policy Manager (APM) functionality works towards simplifying how users and organizations manage access to these assets. The APM through its Rule-Based Access Control (RBAC) model enhances the security protocols of federated authentication, ensuring that only authorized individuals can access specific assets, thereby upholding both security and compliance standards.

The following are the high-level capabilities provided by the Federated AAI and APM subsystems.

Secure Access Control: To ensure that only authenticated and authorized users can access the system or application resources alike the protection of assets.

User Authentication: To verify the identity of users through reliable and secure methods such as SSI, SIOPv2.

User Authorization: To determine and enforce what authenticated users are permitted to do within the system or application based on their roles, permissions, or policies.

User Experience: To ensure that the authentication process is as frictionless as possible without compromising security, thereby improving user satisfaction.

Scalability and Performance: To build authentication and authorization mechanisms that can scale with the growth of users and system demands without degrading performance.

Flexibility and Extensibility: To design the system in a way that supports future expansion, such as adding new authentication methods or integrating with other services.

[1] https://www.w3.org/TR/did-core

[2] https://www.w3.org/2017/vc

1.2 Insights from other Tasks and Deliverables

The purpose of this deliverable is to document the outcomes of Task 3.1 Federated AAI Infrastructure and Task 3.2 Unified Security Policy Management. This deliverable aims to present the FAME Authentication and Authorization Infrastructure (AAI) along with the Assets Policy Manager (APM). As such, this deliverable receives input and refers to work developed as fully open source in i3-MARKET [i3-MARKET] project. The list of the deliverables identified are i3-MARKET's Deliverable D2.1 (Requirements, Specifications, and Co-Creation), and the i3-MARKET's Deliverable D3.2 (Identity and Access Management Specification and Reference Implementation Report). This deliverable primarily serves as first specification and implementation prototype report and as main input for the subsequent deliverables of FAME Work Package 3 (WP3), specifically for Deliverable D3.4 (Secure Federated Data Management II) where more improvements/functionalities can be reported.

1.3 Structure of the Deliverable

The deliverable is structured as follows:

- **Chapter 1 Introduction**: This section serves as the gateway to the deliverable, detailing the main objectives and goals of the document within the context of the FAME project.
- **Chapter 2 Positioning into FAME Solution Architecture**: In this section, the deliverable is contextualized within the larger framework of the FAME Solution Architecture (SA).
- **Chapter 3 Components Specification**: This section delves into the specifics of key components within the FAME project. It includes a comprehensive examination of the SSI Micro Services, starting with a description that outlines their purposes and roles, followed by a discussion of related work and literature. The technical specifications are detailed, along with its interfaces and how they interact with other components. Similarly, the Assets Policy Manager (APM) is explored, beginning with a description, then moving into relevant related work, its technical specifications, and its interfaces within the project. This section helps to understand functionalities of the core components of the FAME project.
- **Chapter 4 Components Demonstration**: This section is dedicated to demonstrating the practical application of the key components discussed earlier. It includes detailed information on the VC and OIDC Service, starting with prerequisites and the installation environment, followed by a step-by-step installation guide and a user guide for end-users. Similarly, the Assets Policy Manager (APM) is covered, providing necessary pre-installation information, an installation guide, and a user manual. This section is instrumental in illustrating how the components are implemented and used within the FAME project.
- **Chapter 5 Conclusions**: The final section of the deliverable encapsulates the key findings, insights, and outcomes derived from the analysis and demonstrations of the FAME SA and its components. It provides a summary of the deliverable, touching on the significant achievements and the Key Performance Indicators (KPIs) met, as outlined in the document.

1.4 Summary of Changes

Table 1 – D3.4 Updates from D3.1

Section	Status	Description of Update/Addition
		Section 1
1.2 - Insights from other Tasks and Deliverables	Updated	Alignment of D3.4 input regarding the technical activities and progress of the project.
1.4 - Summary of Changes	Added	Depiction of updates between D3.4 and D3.1.
		Section 3
3.1.1 - AAI descriptions	Updated	Updated the AAI description according to VII
3.1.2 - Related work	Updated	Added more details
3.1.3 Technical Specification	Updated.	Updated details according to version II, version I details already available in D3.1 and add new C4 diagram
3.1.3 - Interfaces	Updated	Added interfaces for version II
		Section 3.2
3.2.1 - Description	Added	Added Overview of integrated policy manager
3.2.2 - Assets Policy Manager	Updated	Updated details according to version II, version I details already available in D3.1. Added new C4 diagram
3.2.2.1 - Interfaces	Updated	
3.2 - Dynamic Policy Manager	Added	Added section describing Dynamic Policy Manager proof of concept and interfaces
		Section 4
4 - Component Demonstration	Updated	Provided details as per final implemented version, for deployment and user guides
		Section 5
Conclusions	Updated	Update of the overall conclusions of the deliverable to reflect the final findings and outcomes of the deliverable.

2 Positioning into FAME Solution Architecture

In the C4 architecture model of the FAME-SA (Solution Architecture) [A. Mavrogiorgou, et al][FAME-D2.1] (see Figure 1), the Federated Authentication and Authorization Infrastructure is classified as a container named "Authentication and Authorization". This container is included in the "Federation Manager" system. Its role is to provide authentication and authorization using distributed identity and verifiable credentials. To achieve this, it receives asset metadata from the Federated Data Asset Catalogue and supports the "Transaction Operations" system by providing these asset metadata. Additionally, it manages the authorization credentials received from the OpenAPIs.

Furthermore, the Asset Policy Manager is classified as a container named "Assets Policy Management", also included in the "Federation Manager" system. Its primary role is to ensure the secure management of assets within FAME. It interacts with the container named "Regulatory Compliance" to update information regarding applicable regulations. It also exchanges asset policy rules with the data assets catalogue and runs a function for FAME-compliant policy formats.

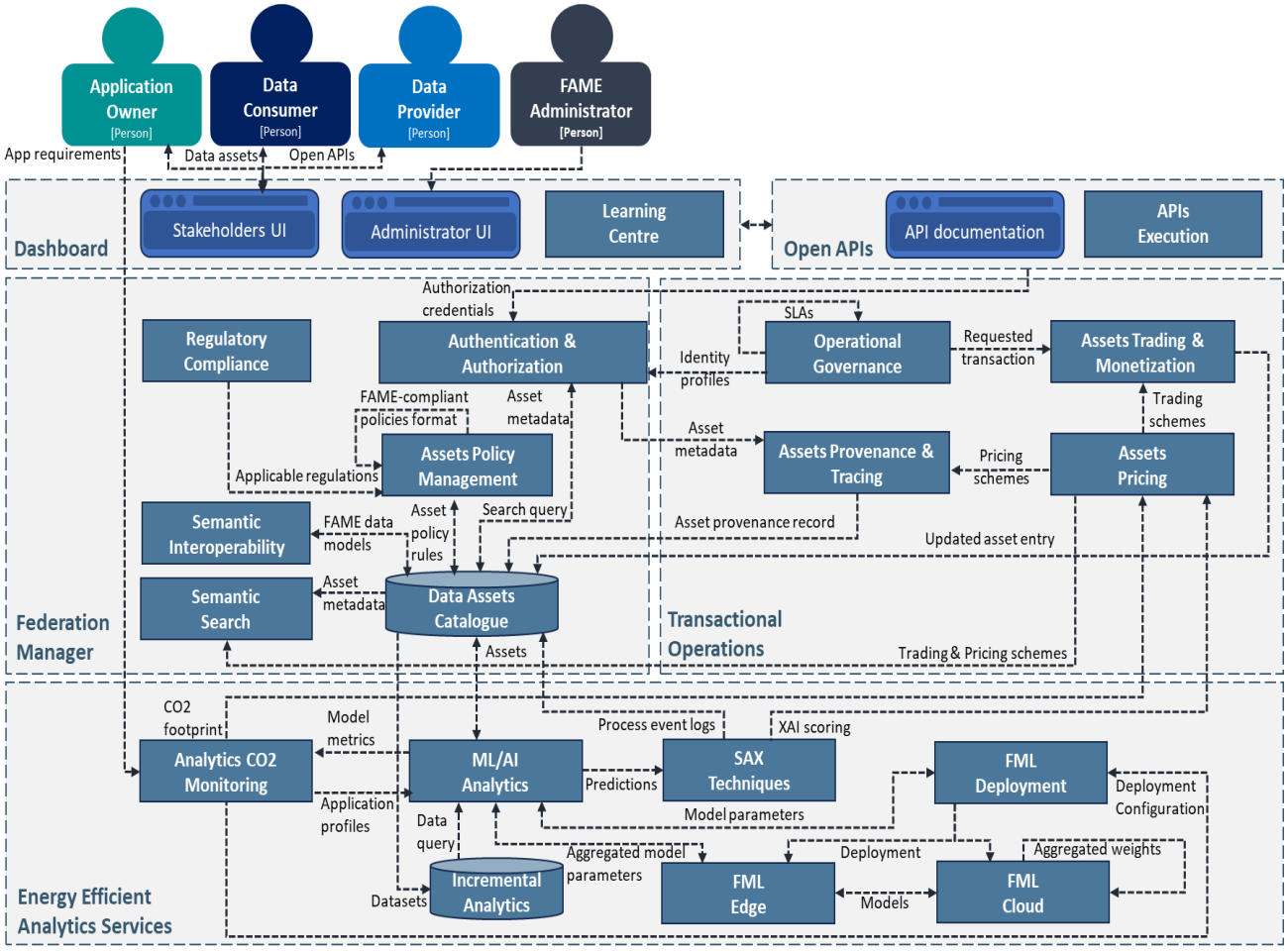

Figure 1 – FAME SA C4 Container Diagram [FAME-D2.1]

3 Components Specification

3.1 Authentication & Authorization Infrastructure (AAI)

3.1.1 Description

The Authentication & Authorization Infrastructure (AAI) is a core component within the FAME system, responsible for managing authentication and authorization processes. Serving as the central authority, AAI ensures secure and seamless access control across all FAME modules. This federated infrastructure provides a comprehensive solution for managing digital identities, verifiable credentials, and access control policies, promoting streamlined and consistent security practices throughout the system. Key capabilities include:

Identity Management:
- Creating, storing, and verifying digital identities and associated credentials
- Issuing, managing, and revoking verifiable credentials

Authentication:
- Verifying the identity of users and entities attempting to access the system.
- Leveraging verifiable credentials for secure authentication

Authorization:
- Controlling access to resources based on verified credentials and defined policies.
- Enforcing fine-grained access controls and permissions

Interoperability:
- Utilizing open standards like OpenID for Verifiable Credential Issuance (OID4VCI) [OpenID-OID4VCI] and OpenID for Verifiable Presentations (OID4VP) [OpenID] for cross-system integration
- Enabling federated identity and access management across different domains

AAI offers a robust, standards-based solution for identity and access management, serving as the backbone of secure access control across the entire system. AAI provides a cryptographically secure mechanism for the issuance and management of Verifiable Credentials, empowering users with enhanced control over their digital identities and permissions. Utilizing Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs), the system enables users to manage their identities independently, reducing reliance on a central authority and strengthening both privacy and trust.

To ensure secure, standardized credential exchange, AAI integrates OID4VCI [OpenID-OID4VCI], facilitating a reliable protocol for identity information exchange. This protocol supports secure, consent-based sharing of credentials, which reinforces user autonomy over personal data while streamlining the verification process for ease of use.

In addition, AAI is designed with rigorous adherence to the latest security protocols and privacy standards, establishing a resilient defence against common vulnerabilities and emerging threats in the digital identity landscape. The platform is built to accommodate current digital verification needs while maintaining flexibility for future advancements in decentralized identity, ensuring that it remains adaptable to the evolving requirements of secure, privacy-preserving access management.

3.1.2 Related Work

As described in the deliverable D3.1, AAI has been developed for the needs of FAME project, utilizing a combination of solutions from relevant project (e.g., i3-Market SSI technology) with open-source tools and technologies. This fundamental component addresses a list of technical and business requirements as these have been described in the Deliverable 2.5 (Analysis, Specifications and Co - Creation II). More specifically, it is designed to address the following objectives:

- **TR002:** Be able to access the assets of several marketplaces and data spaces using a single sign-on mechanism.
- **TR042:** Be able to implement federated authentication and authorization mechanisms to verify users and manage their access to system facilities.
- **TR301:** Be able to have access to resources securely.
- **TR302**: Be able to implement access control within the system.
- **TR303:** Be able to implement OpenID for Verifiable Presentations (OID4VP) protocol for authentication, and the OpenID for Verifiable Credential Issuance (OID4VCI) protocol for user onboarding.
- **TR304:** Be able to use decentralized identifiers (DIDs) to identify data providers and (organizations).
- **TR305:** Be able to implement a system for creating credentials that can be issued, owned, and verified independently of a central authority.
- **TR306:** Be able to implement verifiable credentials and access control using a decentralized system.
- **TR307:** Be able to provide authentication and authorization through distributed identity and verifiable credentials.
- **TR901:** Support interfaces for data assets trading, pricing, and data policy management.

Security in distributed and federated systems is a major concern attracting both academic and industrial focus due to its significant impact on technology adoption. Authentication and authorization are prioritized areas where existing methods often fall short. Typically, deployment systems implement separate methods for authentication and authorization, with few integrated solutions available. The study in [Fayad 2018] introduces an adaptive, blockchain-based approach for both authentication and authorization in IoT, implemented in Java and its extensive evaluations demonstrate the approach effectiveness in meeting various requirements while maintaining low operational costs. Adaptive authentication is a flexible, risk-oriented method of verifying identities that uses up-to-date data to assess the risk associated with a user's actions, dynamically allocates authentication measures based on various factors, and adjusts security measures in response to current events to enhance system security and prevent unauthorized access. Adaptive authentication enhances identity verification by dynamically assessing risks in real-time and adjusting security measures accordingly. It employs risk profiles, which categorize access requests into varying risk levels and determine the necessary authentication methods, such as biometrics or PINs [Shah 2023]. This approach utilizes AI and machine learning to continually refine these assessments and automate decision-making, thereby bolstering security and trust. Adaptive authentication offers a complex solution that surpasses the capabilities of traditional and multi-factor authentication systems. Adaptive authentication leverages various techniques to assess and mitigate risks within an authentication system, enabling decisions on user access based on factors such as location, behaviour, and user attributes. This method integrates dynamic and static elements to tailor authentication requirements according to the sensitivity of the accessed resource. Benefits of this approach include frictionless authentication for trusted entities, enhanced risk management through intelligent assessments, continuously updated security protocols, and flexible policies accommodating Bring Your Own Device (BYOD) environments.

Singh et al. [Singh 2022] introduced the Resilient Risk-based Adaptive Authentication and Authorization (RAD-AA) framework, engineered to dynamically adjust based on risk scores and trust profiles, thereby enhancing system security. They underscored that credential theft has become a predominant attack vector in recent cyber incidents, facilitating unauthorized system access via techniques such as token manipulation. This highlights the critical need for robust authentication and token-based authorization systems. The team evaluated RAD-AA against established standards like OAuth 2.0, OpenID Connect [OpenID-Connect], and SAML 2.0, and assessed its resilience using threat models including STRIDE and PASTA. Furthermore, they investigated the application of machine learning within RAD-AA to precisely evaluate transaction risks, significantly complicating the efforts of adversaries to initiate and sustain attacks on critical infrastructure.

There are several surveys. Bumiller et al. [Bumiller 2023] provide a detailed analysis and overview of research on Context Modelling for Adaptive Authentication systems using Systematic Mapping Study (SMS) and Systematic Literature Review (SLR) methods to identify key properties for these models in adaptive systems. In [Lopez 2004] authors reviewed and evaluated technologies like Microsoft .NET Passport, Liberty Alliance Project, Kerberos-based solutions, and digital certificates along with PKIs for developing AAIs, and concluded that no single technology is superior; each has its pros and cons, and an effective AAI should integrate multiple technologies and strategies. Trnka et al. [Trnka 2022] analysed Authentication and Authorization Advancements for the Internet of Things, categorizing security solutions, IoT security practices, technologies, and standards used in recent research, offering a practical guide and overview of recent research efforts in these areas. Lewis et al. [Lewis 2023] review current Access Control techniques and trends, examining the evolving cyber-attack landscape and zero-trust networking challenges, and discuss the application of Access Control across key domains such as Cloud Computing, Blockchain, IoT, and Software-Defined Networking, and explores business adoption strategies and integration into cybersecurity and network architectures. Ma and Qian [Ma 2024] introduced a scalable identity management scheme using blockchain technology to enhance identity protection and traceability, which integrates blockchain with Reed-Solomon encoding and Certificateless Aggregate Signature to protect user information, with nodes storing parts of the encoded data block. The authors argue that unlike traditional solutions, their approach prevents single points of failure, secures identities, and tracks malicious users, allowing single-upload access across multiple applications without multiple passwords. [Patricia et al. 2019] conducted a survey on Adaptive Authentication, which dynamically selects the most suitable user authentication methods based on contextual factors such as location and device proximity. Despite its potential to revolutionize the prevalent password-based authentication system, practical applications in everyday life are still limited. To improve the design of adaptive authentication, they conducted a comprehensive review of the existing research, identifying current challenges and proposing future directions. Otta et al. [Otta 2023] conducted a systematic survey on Multi-Factor Authentication for cloud Infrastructure, highlighting the urgent need for robust protective measures against unauthorized service use in cloud computing. Central to these measures are enhanced authentication and access control. Traditionally, authentication relied on single-factor methods like usernames and passwords. However, advancements in computing speeds and techniques, from brute force to sophisticated cryptographic algorithms, have exposed vulnerabilities in these systems. As a result, multi-factor authentication, which employs multiple simultaneous authentication factors, has become essential for bolstering cloud security. The survey provides an extensive review of factors that influence the adoption and effectiveness of multi-factor authentication systems, ultimately advocating for a unique, biometric-based authentication factor that does not require specialized equipment, thereby enhancing security and mitigating impersonation risks.

3.1.2.1 Advancements beyond the Related Work

The integration of OID4VCI + OID4VP technology in the FAME project, particularly within the AAI component, signifies a major shift in digital identity verification and data transmission, as highlighted in deliverable D3.1. FAME utilizes issuer and verifier microservices to implement DIDs and VCs, thereby creating a secure, user-centric system. In particular, FAME allows for the separation of concerns between user management and access control. The user management process, implemented by VC issuers that play the role of *Onboarding Authorities* in the FAME federation, is responsible for the protection of Personally Identifiable Information (PII) [IBM-PII] in compliance with current European regulation (e.g., GDPR). On the other hand, the access control function, which is performed by the FAME Platform, is totally shielded from PII and is thus exempt from the burden of personal data regulatory compliance (more details on this architecture are provided later in this document). This method improves data integrity, streamlines credential verification, and adheres to privacy-by-design principles, granting consumers increased control over their data while complying with legal regulations. This integration protects the foundation of the FAME federation, promoting seamless collaboration and confidence among stakeholders.

Although many current solutions often follow the SSI paradigm, FAME's AAI module surpasses prior efforts by leveraging the Sphereon software development kit (Sphereon SDK) [Sphereon-SSI-VC]. This use offers sophisticated capabilities, like selective disclosure and zero-knowledge proofs for privacy-preserving authentication, as well interoperability with many SSI ecosystems, most notably the European Digital Identity (EUDI) framework [N. Sakimura, J. Bradley and N. Agarwal 2015].

Regarding the use of the standards, AAI innovates to improve the general security. While most SSI implementations use JavaScript Object Notation (JSON) Web Tokens (JWT) [Sphereon-Wallet], AAI makes use of an innovative token lifetime management strategy to guarantee tamper resistance and prompt revocation. This function reduces possible weaknesses in authorization flows, therefore posing a shared difficulty in comparable tasks using JWTs without revocation policies.

To summarize, FAME's Authentication & Authorisation architecture not only meets project needs but also establishes a new norm for safe and effective distributed identity solutions by tackling interoperability, privacy, and automation.

3.1.2.2 The Imperative for Adaptive Trust Interoperability in Federated Data Spaces

A considerable volume of research has been dedicated to the domain of trust interoperability within data spaces. Notable studies include Semantic Interoperability in Resilience Data Spaces by Gutt [Gutt 2024], Trust Interoperability in IoT by Fernandez-Gago [Fernandez-Gago 2024], and Interoperability of Heterogeneous Trust Schemes by More [More 2023]. Additionally, investigations into Self-Sovereign Identity for Trust and Interoperability by Ghirmai [Ghirmai 2022], Secure and Trusted Privacy-Protected Data Exchange by Cuñat [Cuñat 2024[, and Robust Trust Frameworks for Data Exchange by Badirova [Badirova 2024] have significantly advanced the field.

These scholarly works highlight the multifaceted challenges enterprises face when deploying diverse tools and approaches, leading to a fragmented security landscape. Such fragmentation diminishes visibility and exacerbates security risks, thereby stressing the urgent need for enhanced interoperability among security solutions. This becomes particularly critical as organizations strive to meet approaching deadlines for implementing zero-trust architectures, which necessitate the integrated functioning of security tools to consistently verify the legitimacy of users and devices.

In the context of federated data spaces—where data sharing extends across multiple organizational boundaries—the demand for adaptive trust interoperability is paramount. These settings require a sophisticated framework that not only supports efficient data exchange but also upholds rigorous security and compliance with varied regulatory norms. Central to this framework is adaptive trust interoperability, which introduces a dynamic method for managing trust and security in a constantly evolving threat environment.

The Continuous Adaptive Trust (CAT) model is especially pertinent in these federated data spaces. It emphasizes continuous, adaptive, and trust-based access control, extending beyond conventional security measures like Multi-Factor Authentication (MFA) by persistently evaluating user risk post-initial authentication. This ongoing assessment is crucial in federated settings where the sensitivity of data can vary, and the assurance of user authenticity needs relentless verification.

Further enhancing the resilience of federated data spaces against sophisticated threats is the integration of adaptive risk-based security with trust-based security, as suggested by Abie et al. [Abie 2022]. This blended approach facilitates a more nuanced response to security challenges, dynamically adjusting trust levels based on real-time risk evaluations and behavioural analytics. Such adaptability is vital in environments where interactions and data exchanges span across disparate systems with potentially different security postures.

Moreover, the deployment of a variety of metrics and tools to assess and ensure the trustworthiness of systems engaged in a federated data space is crucial. This thorough evaluation helps maintain a high standard of security integrity throughout the data exchange process, ensuring that all entities involved meet the requisite trust and security standards.

The practical implementation of adaptive trust interoperability in federated data spaces significantly strengthens security measures, rendering the data exchange process more secure and effective. By dynamically modifying security policies and trust levels based on continuous risk assessment, federated data spaces are better equipped to safeguard sensitive data against unauthorized access and potential breaches. This strategy not only fortifies the security framework but also establishes a robust foundation of trust among participating organizations, which is essential for successful and enduring data collaborations in federated settings.

3.1.3 Technical Specification

3.1.3.1 Component-level C4 Architecture

The Authentication & Authorization Infrastructure (AAI) is a scalable, secure, and interoperable solution designed to support distributed identity management using verifiable credentials. Built on a microservices architecture, AAI comprises two primary services and one mobile app: the **Verifiable Credential Issuer Service**, the **Verifier Service** and the **Identity Wallet**, respectively**.** Together, they handle the issuance and verification of digital credentials, enabling secure and trusted interactions in a decentralized digital ecosystem.

- Microservices-based architecture, ensuring modularity, scalability, and independent deployment of services.
- Supports OpenID Connect (OIDC) and OAuth 2.0 [N. Sakimura, J. Bradley and N. Agarwal 2015] protocols to enable secure and standardized communication across services.
- Supports the OID4VP and OID4VP protocols to enable SSI workflows.
- Compliant with W3C standards for Verifiable Credentials (VC) and Decentralized Identifiers (DID), supporting cross-platform compatibility and interoperability.

The Verifiable Credential Issuer Service and the Verifier Service, although being both part of the AAI module, have different positionings within FAME's architecture. VC issuance is done by Onboarding Authorities (OAs), which are organizations that, being approved members of the FAME federation, take on responsibility for user management and *onboarding* – i.e., the process that enables a person to actively participate in the FAME ecosystem. In practice, users are typically the employees and/or customers of an OA, meaning that every user is known to their OA of provenance and that such OA has the means to verify their real identity. The onboarding process for a user is complete when the newly onboarded user receives a FAME Verifiable Onboarding Credential (VOC), thus becoming the holder of an OA-certified digital identity. VOCs are issued by the Verifiable Credential Issuer Service, *which is run and operated autonomously by the OA*. On the other hand, the Verifier Service is an integral part of the FAME Platform for the trading operations: users need to present a valid VOC to the Verifier Service to access restricted content.

The Identity Wallet mobile app, installed on the user's personal device, acts as a mediator between the two services, as shown in the diagrams in Figure 2 and 3.

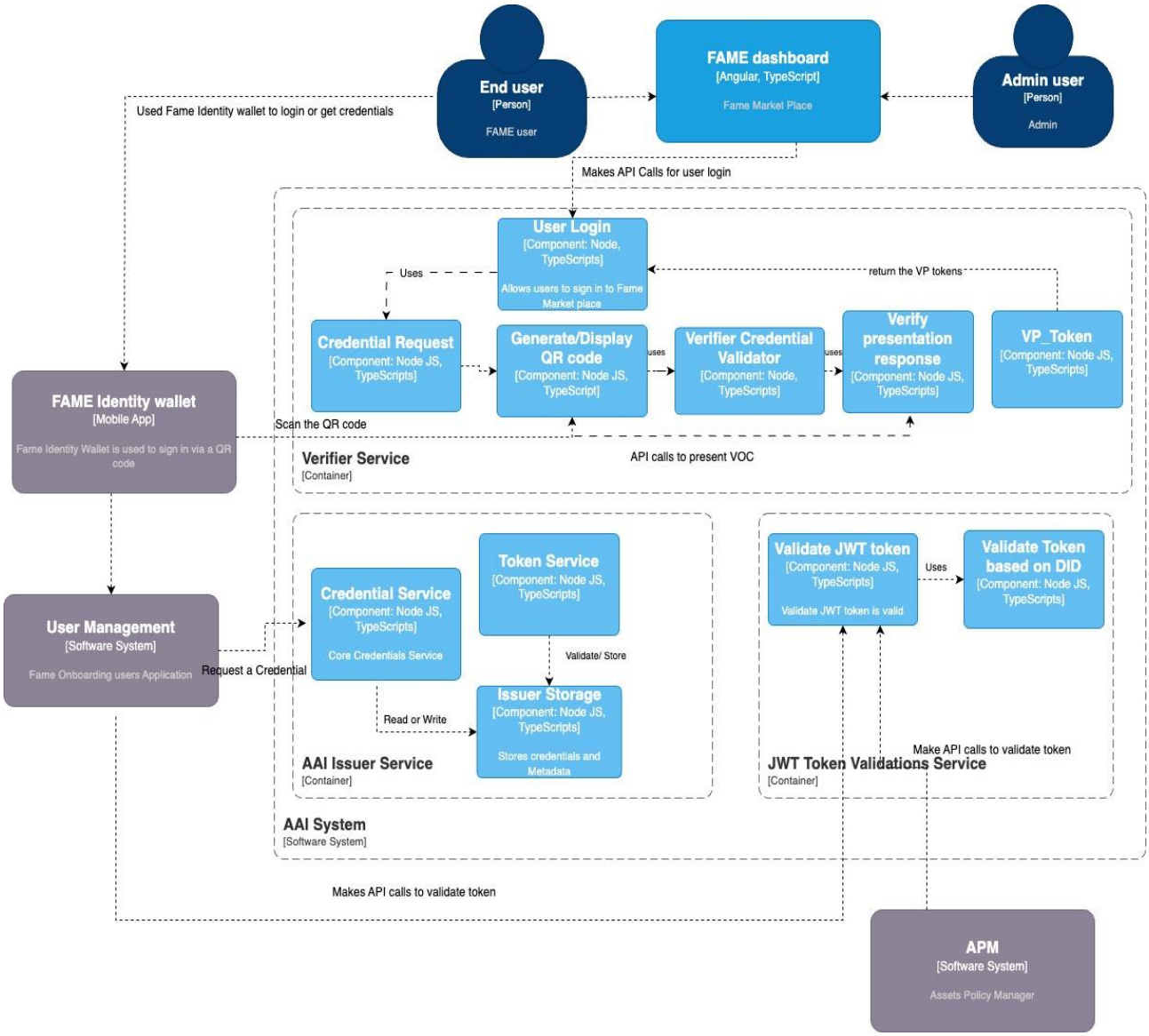

Figure 2 – AAI C4 Container Diagram

3.1.3.2 Verifiable Credential Issuer Service

- **Purpose**: To provide secure issuance of VOC credentials through standardized APIs.
- **Key Functions**:
    - **Credential Generation**: Creates credentials in compliance with W3C standards, allowing for proof of identity or attribute claims.
    - **Credential Formatting**: Structures credentials to be portable and readable by various verifiers across ecosystems.
    - **Credential Issuance**: Issues digitally signed credentials to users, ensuring data integrity and authenticity.

In reference to the Issuer Service process shown in Figure 3, it outlines the workflow for creating and managing verifiable claims. These claims are subsequently embedded in the ID token and can be utilized during authentication flows for verification purposes.

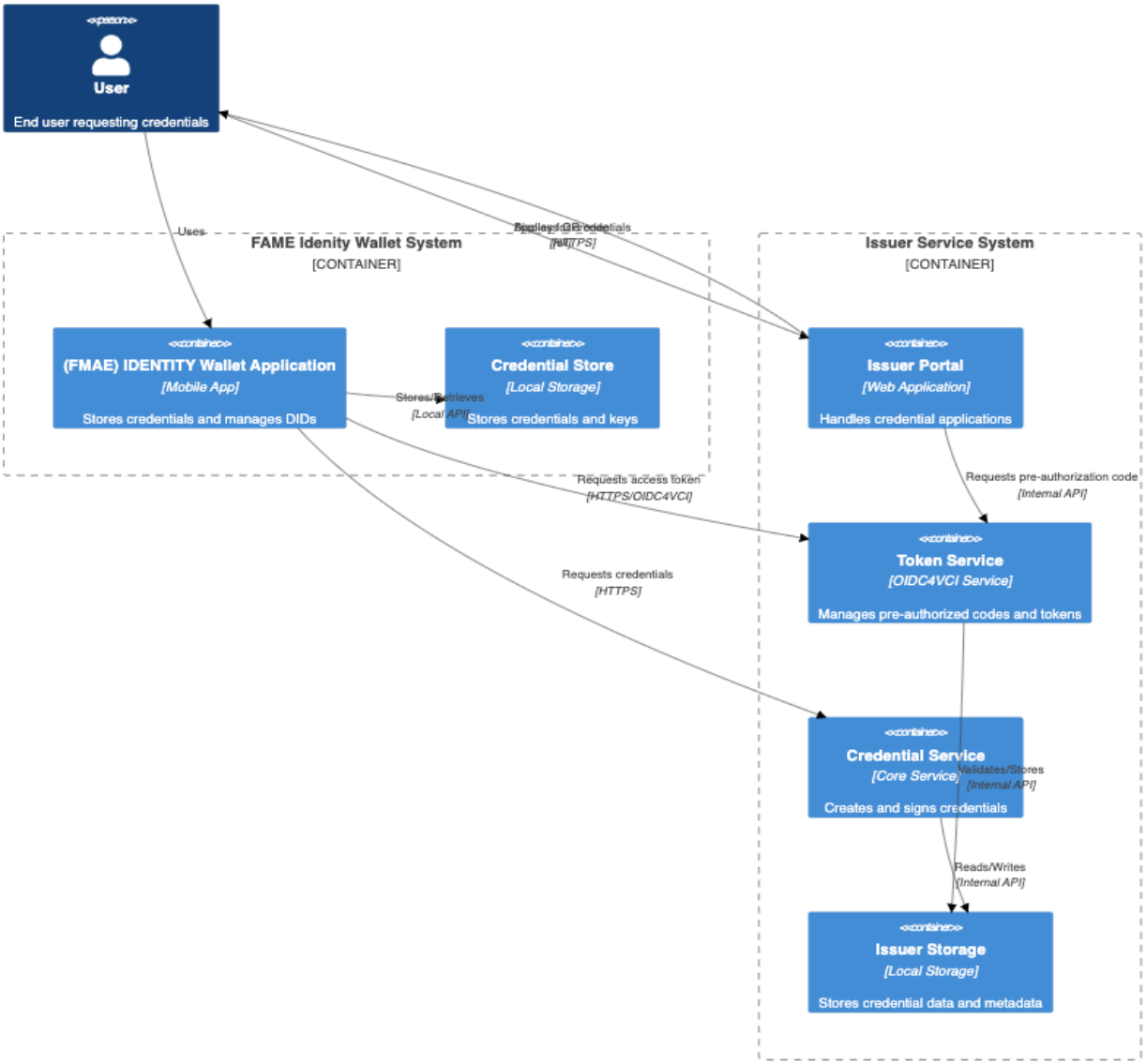

Figure 3 – Verifiable Credential Issuer Service: Issuance Process

3.1.3.3 Verifier Service

- **Purpose**: To validate user-presented VOC credentials, ensuring authenticity and integrity before granting access or permissions.
- **Key Functions**:
    - **Credential Verification**: Verifies that the user-provided VOC credential originates from a trusted OA issuer, is still valid and has not been tampered with.
    - **vp_token Generation**: Issues to the user a Verifiable Presentation Token (vp_token) as proof of successful verification, enabling access to resources.

The Verifications service process, as depicted in Figure 4, outlines the steps involved in confirming the validity and authenticity of a user-provided VOC (Verifiable credential. This process ensures that the credential originates from a trusted OA (Organization Authority) issuer, thereby establishing its reliability. Additionally, the service authenticates the user during the verification process, ensuring that the individual presenting the credential is its rightful owner. This dual-layer verification enhances the security and trustworthiness of the system.

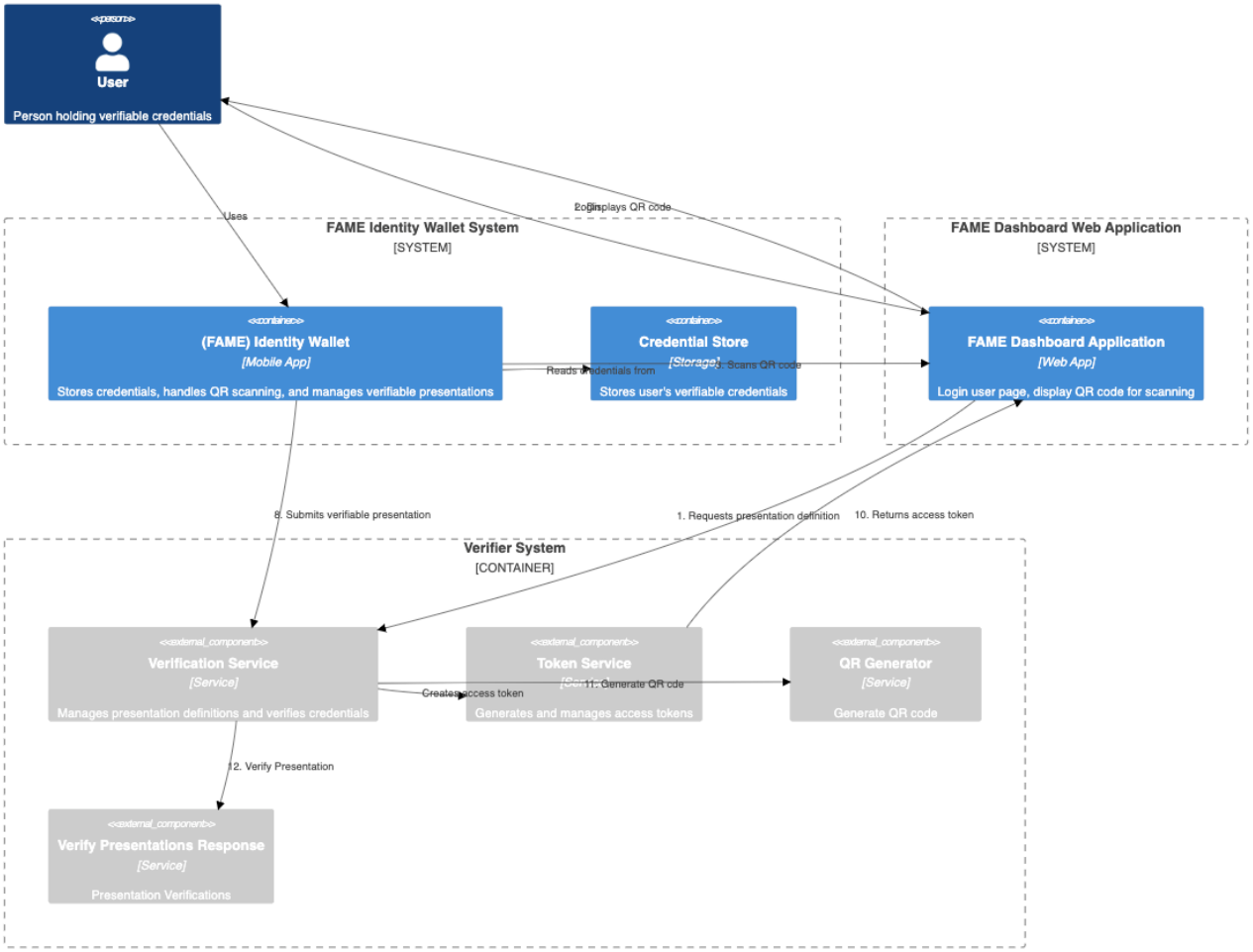

Figure 4 – Verification Service : Login Process

3.1.3.4 Identity Wallet Mobile App

- **Purpose**: Act as the user-side client, enabling the user to participate in both the VOC issuance and verification processes.
- **Key Functions**:

    - **Credential Retrieval**: Retrieves an issued VOC from a Verifiable Credential Issuer Service.
    - **Credential Secure Storage**: Maintains retrieved VOCs in an encrypted database on the user's personal device.
    - **Credential Presentation**: Presents a VOC stored on the user's personal device to a Verifier Service.

The user-side client enables seamless participation in the VOC issuance and verification processes. It retrieves VOCs from the Verifiable Credential Issuer Service, securely stores them in an encrypted database on the user's device (FAME Identity wallet) and facilitates their presentation to a Verifier Service for authentication.

3.1.3.5 Baseline Technologies and Tools

The Federated Authentication & Authorization Infrastructure (FAAI) was implemented following the Free Open Source Software design principles and using a range of open-source technologies and tools to ensure scalability, security, and interoperability. This combination of tools and technologies enabled a robust, agile, and secure implementation of the AAI system.

- Key JavaScript libraries such as @sphereon/ssi-sdk-core [Sphereon-SSI-VC] and @sphereon/did-auth-siop [Sphereon-SSI-VC] were employed for decentralized identity and authentication management.
- Core programming was carried out using JavaScript and TypeScript on the Node.js runtime, while JSON Web Tokens (JWT) [Sphereon-Wallet] were used as a standard for secure data exchange, facilitating authorization flows and token-based communication between services.
- The Identity Wallet mobile app was developed from an existing open-source SSI wallet provided by the Sphereon community [Sphereon-SSI-VC], originally developed as a React Native app: in the AAI context, it was modified to better support the FAME onboarding and login processes and rebranded for use within the FAME ecosystem. For managing digital credentials,
- AAI leveraged W3C's Verifiable Credentials (VC) standard, providing secure and interoperable identity management, along with Decentralized Identifiers (DIDs) [W3C-DIDs] to create private and user-controlled digital identities.
- To support DevOps and CI/CD, AAI used Docker for containerizing microservices, ensuring consistency across environments, and Kubernetes [Kubernetes] for orchestrating these containers, allowing automated deployment, scaling, and management.
- Continuous integration and deployment were automated through tools like Harbor [Harbor], Argo CD [Argo-CD] and GitLab [GITLAB] CI, streamlining the build, test, and deployment processes.

- Git versioning control and team collaboration was utilized to track code changes, while GitLab provided a platform for repository hosting, code review, and project collaboration.
- React JS was used for building standalone demonstrators/prototypes i.e. Version 1 and for demo purposes including the front end [React JS], and
- Angular JS is used in the process for building and deploying the integrated version i,e.Version 2 with FAME dashboard [Angular JS].

3.1.3.6 Issue a Verifiable Credentials

In the innovative registration process at the Data Marketplace, the user is issued a Verifiable Credential, marking the commencement of their engagement (Figure 5). Here, a specialized microservice first authenticates the user's Decentralized Identifier (DID) [W3C-DIDs]. Upon successful authentication, it issues a Verifiable Credential linked to this DID, certifying the user's designated role within the marketplace, which could be as a data consumer, data provider, or a combination of both. The user interaction process is defined in the following C4 component diagram, which illustrates how the user can obtain the VOC.

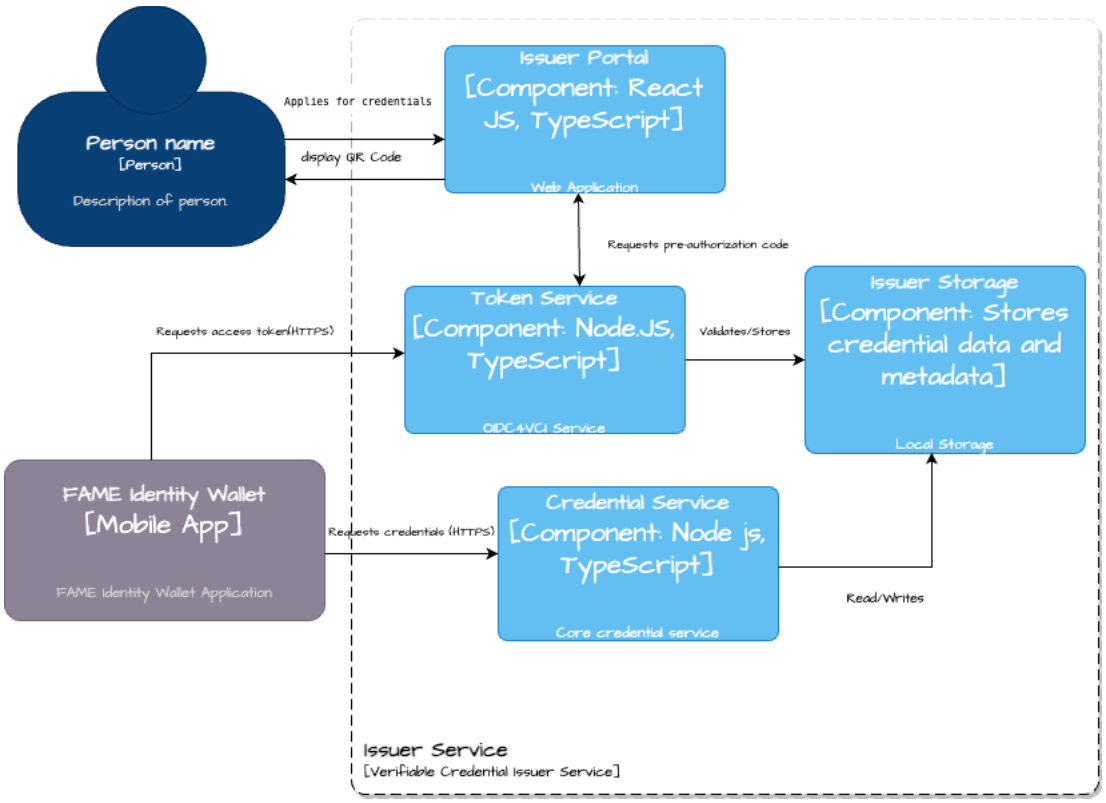

Figure 5 – Issuer Service C4 diagram

The interactions and multiple activities involved in this process are delineated in the following sequence diagram (Figure 6), which is the generalisation of the user registration process. The purpose of including this generalisation diagram is to clearly outline the roles and interactions of the various entities involved in the user registration.

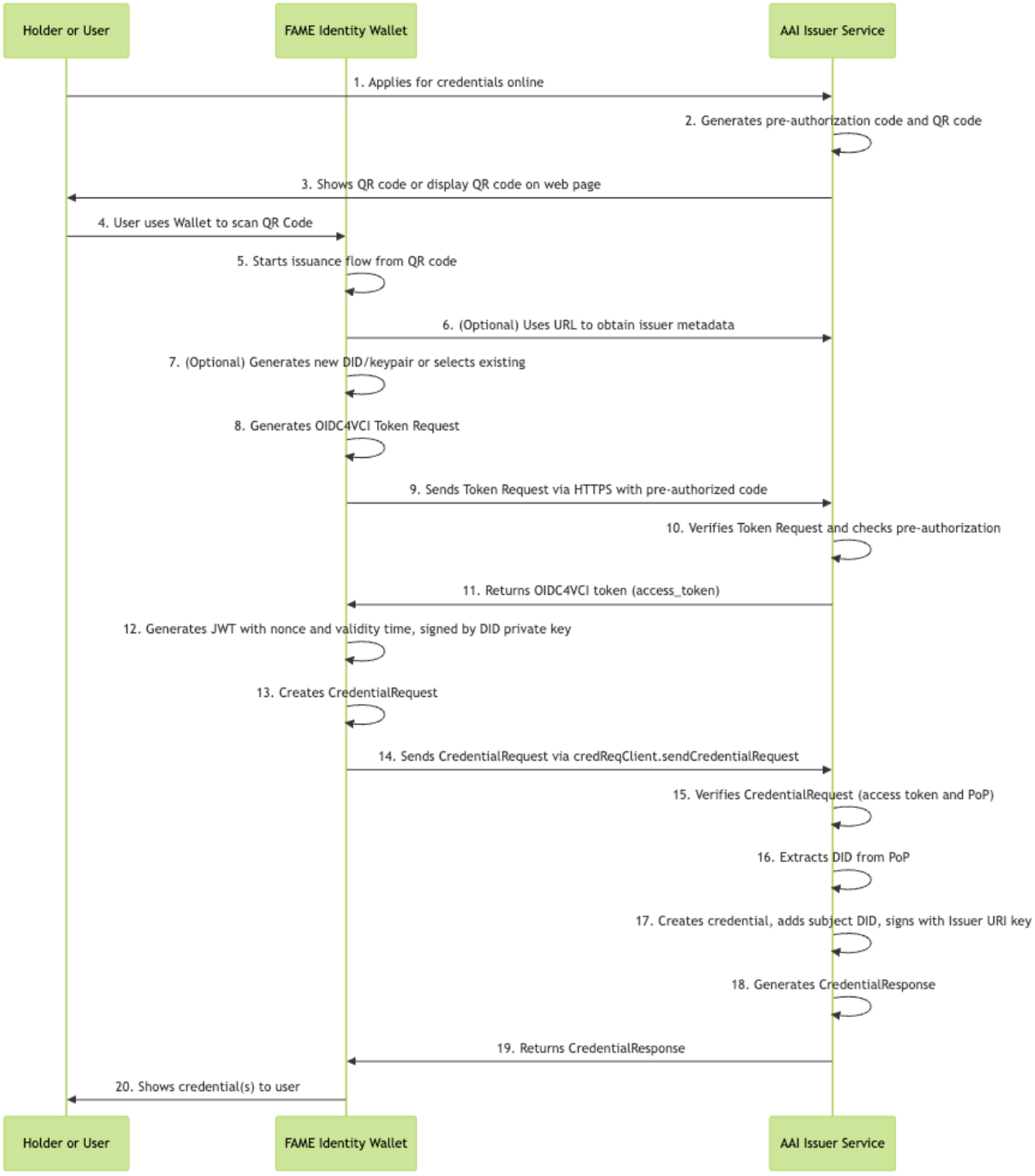

Figure 6 – Credential issue sequence diagram

Initial Authorization Phase:

- The Issuer generates a pre-authorization code with limited validity.
- This code is encoded into a QR format containing an Issuance Initiation Request
- The QR code serves as a secure bridge between the web-based application process and the mobile wallet environment.

Token Exchange Process:

- The Wallet scans and processes the QR code to extract the pre-authorization parameters.
- It may optionally fetch the Issuer's metadata to verify the endpoint authenticity.
- A new DID (Decentralized Identifier) [W3C-DIDs] or keypair can be generated, or an existing one selected.
- The Wallet constructs an OIDC4VCI Token Request containing the pre-authorization code.
- This request is transmitted securely via HTTPS to the Issuer's token endpoint.

Credential Issuance Phase:

- Upon successful token verification, the Issuer returns an access token.
- The Wallet generates a JWT (JSON Web Token) [IETF-RFC7519] containing a nonce and short validity period.
- This JWT is signed using the DID's private key, providing Proof of Possession (PoP)
- A formal CredentialRequest is created and sent via a dedicated client function.
- The Issuer performs comprehensive verification of both the access token and PoP.
- The subject's DID is extracted and embedded into the credential.
- The final credential is signed using the Issuer's private key associated with their URI.

The implementation relies on several crucial components:

1. Wallet System Components:
    - Mobile Wallet Application: Handles user interactions and credential management
    - Secure Storage: Manages credential and cryptographic key storage.
    - Cryptographic Module: Performs key generation, signing, and verification operations.

2. Issuer Infrastructure:
    - Web Portal: Manages credential applications and user interactions.
    - Token Service: Implements OIDC4VCI protocol specifications.
    - Credential Service: Handles credential generation and signing.

Security Considerations

The protocol implements multiple security measures:

- All network communications occur over HTTPS.
- Time-limited tokens and nonces prevent replay attacks.
- Proof of Possession mechanisms ensure credential binding to the intended recipient.
- DID-based signatures provide cryptographic proof of credential origin.
- Segmented storage systems protect sensitive key material.
- Pre-authorization codes provide a secure bridge between web and mobile environments.

1. **Identity Holder (FAME user):** The individual or entity associated with a specific identity within the FAME system, representing a user who holds verifiable credentials and interacts with the platform.
2. **FAME Identity Wallet**: A secure digital storage facility where verifiable credentials are stored for the FAME user. It serves as a repository for authentication and authorization-related information within the FAME ecosystem.
3. **Issuer Verifiable Credential Microservice (FAME Data Marketplace instance):** A specialized service within the FAME Data Marketplace that handles the generation, verification, and management of verifiable credentials. It plays a crucial role in ensuring the integrity and security of identity-related information.

3.1.3.7 Verify a Verifiable Credential or User Login process

The FAME AAI framework employs cutting-edge OpenID Connect standards, specifically OID4VP (Verifiable Presentations) and SIOPv2 (Self-Issued OpenID Provider), to deliver a sophisticated, secure digital identity verification system. This approach transforms credential exchange by enabling seamless, cryptographically robust transport of Verifiable Credentials through standardized OpenID Connect protocols. Relying Parties gain a powerful mechanism to issue, validate, and manage digital credentials with unprecedented trust, interoperability, and security.

The FAME AAI framework enables a decentralized Identity and Access Management (IAM) system that transcends traditional centralized models. By implementing OpenID Connect (OIDC) and W3C Verifiable Credentials standards, we create a robust, resilient authentication infrastructure that ensures fault tolerance, enhanced security, and unprecedented trust without dependency on centralized Identity Providers (IdPs)

By leveraging OpenID Connect (OIDC) for Verifiable Credentials transport, our Decentralized Identity and Access Management Framework enables sophisticated, data-driven authorization mechanisms. This approach empowers participants to implement advanced access control paradigms—including granular Role-Based Access Control (RBAC), dynamic Attribute-Based Access Control (ABAC), and comprehensive policy enforcement—by seamlessly integrating attested credential data into authorization decisions.

The diagram in Figure 6, depicts a comprehensive authentication workflow, involving interactions between the user, login server, and verification services. Key steps include credential presentation, identity verification, token issuance, and secure session establishment.

This detailed sequence illustrates the robust, distributed nature of the authentication system, leveraging standards-based protocols and decentralized validation mechanisms to ensure secure, trusted access management.

The visualized process showcases the system's resilience, flexibility, and adherence to industry best practices, delivering a scalable, fault-tolerant identity and access management solution.

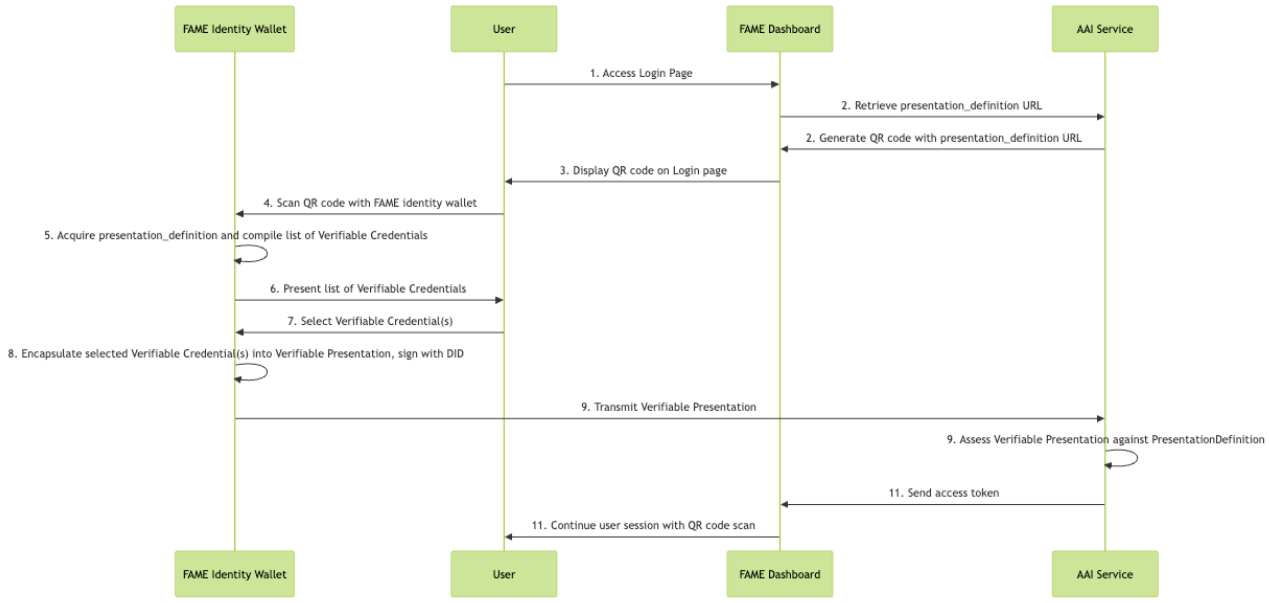

Figure 7 – Login process sequence Diagram

Based on the sequence diagram figure 7, here are the technical points for each step:

1. **User Accesses Login Page**
   - The user initiates the login process by accessing the login page of the Fame Dashboard.
2. **Fame Dashboard Requests Presentation Definition URL**
   - The Fame Dashboard requests a presentation definition URL from the API Service, defining the required attributes for authentication using Verifiable Presentations.
3. **API Service Generates Presentation Definition URL**
   - The API Service responds with a presentation definition URL. This URL conforms to OID4VP and is structured to request specific Verifiable Credentials from the user's wallet.
4. **Fame Dashboard Displays QR Code**
   - Using the presentation definition URL, the Fame Dashboard generates and displays a QR code. This QR code facilitates the OID4VP flow by allowing the user to scan it with their digital wallet.
5. **User Scans QR Code with Digital Wallet**
   - The user scans the QR code using a digital wallet application, initiating a request to present the required Verifiable Credentials.
6. **Identity Wallet Processes Presentation Definition**
   - The digital wallet interprets the presentation definition from the QR code and checks if the necessary Verifiable Credentials are available within the wallet.
7. **User Selects Verifiable Credentials**
   - The user selects the appropriate verifiable credentials (e.g., ID) within the digital wallet.
8. **Identity Wallet Encapsulates and Sends Verifiable Presentation**
   - The wallet encapsulates the selected Verifiable Credentials into a Verifiable Presentation, structured per the OID4VP protocol, and sends it to the Fame Dashboard.
9. **Fame Dashboard Transmits Verifiable Presentation to API Service**
   - The Fame Dashboard forwards the received verifiable presentation to the API Service for verification.
10. **API Service Validates Presentation**
    - The API Service verifies the Verifiable Presentation against the presentation definition, ensuring it adheres to the OID4VP and SIOPv2 standards. It confirms the authenticity and validity of the credentials.

11. **API Service Responds with Access Token**
    - Upon successful validation, the API Service generates and sends an access token to the Fame Dashboard, which confirms the user's authentication status.
12. **Field Dashboard Grants User Access**
    - The Fame Dashboard grants access to the user by utilizing the received access token, allowing them to proceed with authorized actions.

These points break down the sequence of events in terms of interactions between the user, digital wallet, Fame Dashboard, and API Service, highlighting the key technical processes involved in this verifiable credential-based login process.

Table 2 – Baseline Technologies and Tools

Baseline Technology	Description	Added value to FAME
i3-MARKET SSI	Identity framework	i3-MARKET already performed a research work and implemented SSI, this technology helping to establish a decentralized authentication and authorization solution for the FAME marketplace.
Sphereon Wallet	SSI wallet mobile app	This standard-based implementation of an SSI-enabling mobile app for the exchange of Verifiable Credentials was used as the foundation of the FAME Identity Wallet component.

3.1.4 Interfaces Version I

This section describes the classes and endpoints of the defined interfaces. Verifiable Credentials (VC) service acts as an issuer role in the self-sovereign identity model. The tables included contains the details of the end point descriptions and Components URI alike the API body examples.

Table 3 – issue credentials

FAME TECHNICAL INTERFACE	
Technical Interface ID	#FAAI-1
Endpoint Name	Issue a credentials
Endpoint Description	This APIs produces an HTML page that must be displayed on a browser. It is therefore necessary to redirect the user at the url of this API. The generated HTML page contains a script that communicates with the FAME wallet
Component	VC
Endpoint URI	/issue/{credential}/callbackUrl/{callbackUrl}
HTTP Method	GET
Request (Query) Parameters	{ Consumer: true

	}
Request Body (example)	N/A
Request Headers	N/A
Response Body	Produce HTML page, then rendering of this page, a notification will appear on the wallet asking you to select the identity (DID) on which to save the verifiable credential. After selecting it, it automatically generates the credential and sends it back to the FAME wallet. The wallet will present a new notification asking if you want to accept the credential. Upon acceptance, the credential will be saved and viewable in the wallet.
Response Status Code	200 – OK

Table 4 – Issue a Revoke Credentials.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#FAAI-2
Endpoint Name	Revoke credential
Endpoint Description	This APIs produces an HTML page that must be displayed on a browser. It is therefore necessary to redirect the user at the url of this API. The generated HTML page contains a script that communicates with the FAME wallet
Component	VC
Endpoint URI	/credential/revoke
HTTP Method	POST
Request (Query) Parameters	N/A

Request Body (example)	{
	
	"credentialJwt": "string"
	
	}

Request Headers	N/A
Response Body	Produce HTML page, then rendering of this page, a notification will appear on the wallet asking you to select the identity (DID) on which to save the verifiable credential. After selecting it, it automatically generates the credential and sends it back to the FAME wallet. The wallet will present a new notification asking if you want to accept the credential. Upon acceptance, the credential will be saved and viewable in the wallet.
Response Status Code	200 - OK

Table 5 – Verify a Credential.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#FAAI-3
Endpoint Name	Verify credential
Endpoint Description	This APIs produces an HTML page that must be displayed on a browser.
Component	VC
Endpoint URI	/credential/verify
HTTP Method	POST
Request (Query) Parameters	N/A

Request Body (example)	{
	
	"credentialJwt": "string",
	
	"credentialIssuer": "string"
	
	}

Request Headers	N/A
Response Body	Produce HTML page, then rendering of this page, a notification will appear on the wallet asking you to select the identity (DID) on which to save the verifiable credential. After selecting it, it automatically generates the credential and sends it back to the FAME wallet. The wallet will present a new notification asking if you want to accept the credential. Upon acceptance, the credential will be saved and viewable in the wallet.
Response Status Code	200 - OK

Table 6 – Login to get token.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#FAAI-4
Endpoint Name	Login to get token
Endpoint Description	The first step to use the OIDC provider is to login and get a JWT token to register a new client in the provider.
Component	OIDC node
Endpoint URI	/developers/login
HTTP Method	GET
Request (Query) Parameters	Username & password
Request Body (example)	N/A
Request Headers	N/A
Response Body	{

	"initialAccessToken": "JmmRhudSc6VkVzvIQamP-pe5Zj4k7TR_oqe6uzIbiYW" }
Response Status Code	200 - OK

Table 7 – Client Registrations.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#FAAI-5
Endpoint Name	Client Registration
Endpoint Description	The first step to use the OIDC provider is to register a new client in the provider using JWT token.
Component	OIDC node
Endpoint URI	/oidc/reg
HTTP Method	GET
Request (Query) Parameters	N/A

Request Body (example)	{
	"grant_types": [
	"authorization_code"
	],
	"token_endpoint_auth_method": "client_secret_jwt",
	"redirect_uris": [
	"http://localhost:3000/api/credential"
	],
	"post_logout_redirect_uris": [
	"http://localhost:3000/auth"
	],
	"client_name": "fame_demo_app_05",
	"id_token_signed_response_alg": "EdDSA"
	}

Request Headers	N/A
Response Body	{ "application_type": "web", "grant_types": ["authorization_code"], "id_token_signed_response_alg": "EdDSA", "post_logout_redirect_uris": ["http://localhost:3000/auth"], "require_auth_time": false, "response_types": ["code"],

	"subject_type": "public", "token_endpoint_auth_method": "client_secret_jwt", "introspection_endpoint_auth_method": "client_secret_jwt", "revocation_endpoint_auth_method": "client_secret_jwt", "require_signed_request_object": false, "request_uris": [], "client_id_issued_at": 1703799657, "client_id": "HHn7wzoY_v_zWd-zht3yj", "client_name": "fame_demo_app_05", "client_secret_expires_at": 0, "client_secret": "FXrl6dePLjdVUNp7qNg0UWThBHBEOvsvi7OGV6qONd9CZmpueYssOm-KpayQeiDxXBStqeCG4TJdTc4WoW6ohg", "redirect_uris": ["http://localhost:3000/api/credential"], "registration_client_uri": "https://identity.fame-horizon.eu/release2/oidc/reg/HHn7wzoY_v_zWd-zht3yj", "registration_access_token": "w860m6GOI_4S5GV-18bNfZsAqLvHv5IQem1n-r0mXla" }
Response Status Code	201 - Created

Table 8 – Authentication and Authorization.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#FAAI-6
Endpoint Name	Authentication and Authorization
Endpoint Description	The authorization code flow involves calling two different APIs
Component	OIDC node
Endpoint URI	/oidc/auth /oidc/token
HTTP Method	GET, POST
Request (Query) Parameters	Scope, response_type, client_id, redirect_uri, state, code_challenge
Request Body (example)	N/A
Request Headers	N/A
Response Body	As you can see, this first API returns an HTML page as a response. This page is the login page to be rendered by the user's browser. This means that the user must be redirected to this page via a redirect. Under the hood, this page contains scripts that allow pairing with the wallet. In particular, if a valid pairing is not detected, an automatic procedure is activated. A form is then shown in which the user must copy a code generated through the UI of the i3-Market desktop wallet. Once the pairing code has been entered correctly, the procedure continues automatically, and a notification will appear in the wallet to allow the disclosure of the credentials.

	The second API is the POST /token. You can have a look to the swagger references here. The POST /token is required to exchange an authorization code (previously generated by the GET /auth). As a result it provides an access_token and an id_token, which contains the verifiable credentials revealed during authentication. These credentials contain information about the user.
Response Status Code	200 - OK

Table 9 – JWKS.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#FAAI-7
Endpoint Name	JWKS
Endpoint Description	JWKS endpoint containing the public keys used by OpenID connect relying party to verify any JWT issued by the authorization server.
Component	OIDC node
Endpoint URI	/oidc/jwks
HTTP Method	GET
Request (Query) Parameters	N/A
Request Body (example)	N/A
Request Headers	N/A
Response Body	{"keys":[{"e":"AQAB","n":"0Ycmwhk49Zqnh-nVRyTr-VICexhAklWihwu8YCvQWmASB_almhouFVzuJOtPN-3H2izmzge9zXl9jrDfijc2heOKb6VUogRVdI85FnrwNqWIrxmpzaweQkiCAjjlPmJ2-vIrNffB_cGtRgOauChonPY5cBAYfyzeR9b9eEx3r-R8p-ueu-TPaHH95gh7VIbDKUkGnFP0RFejjhP2vkfcQlhrCt1n8qWjiCrFLYRgeSI-iS2jomYxVMMwvupv1YEalkBeM2nmF0dtuV9ga7yfFZuqip5xC4Fg1oJENvJZ_XKkoQktxobvW4dMz-gbZvVeSQI0ZtjYMK2PqzzosM9BdQ","kty":"RSA","kid":"VpPJ0p_aILtDIhl3Jw7ZNqRJY_cALQAHe4rwbZJTWNE","use":"sig"},{"e":"AQAB","n":"moUHQzhodwqUt8g66nWjeHHeEo--OLg2o38JRVZqInUvQBd0YLroTE0cNcDgsvzGAOjobJ9phEtF_t9G5ppAnXRquLYltBv55ev6uTnGPVPUwb4WQPJhp12lKPhw1uTFlH6vNLK-D1UiaqSi0aE_IJLh1QgHFCp2MTyFgOmigPhfM3QGc_c8sSbDRSylY-SskMk9wU1vpMCjuD1gLcoxboljk2Dhe9uz2Mke-lhy0AFZdJ1K_xu8HkjLunjKMoz3Dtm1c917z3SyqA22l2Az1BeMyPXQaVSsTXCx48FHk8D9lCU1mT0K9ym5JQJxwGiTsapA6J5N4MkypWYEg7JECQ","kty":"RSA","kid":"aL062tC9RFzrqedyRxVR-G0TlZRpXPejpClNvLHvT4I","use":"sig"},{"crv":"P-256","x":"mf4VFYC5AxDVHCaGfwiCdLsbZlbOOdDu4bMLhL54Z9Y","y":"vW6edIo4q-ulhogfAgcCvp6Oa8Cp5_0HNxI2oaVLt6o","kty":"EC","kid":"XOAMa3HgSkc3MtREGUyCN2kjL19kbefx6xmqgnyQbws","use":"sig"},{"crv":"Ed25519","x":"yzGDb7GoVZ2aH8-owgGDlBbgmX8hHPMakSvo7fs9Se8","kty":"OKP","kid":"zR4lbLbFyFKIRMg4u-bwZ35Yr-hd-vXEKQ9BHjnPcos","use":"sig"}]}
Response Status Code	200 – OK

3.1.5 Interfaces Version II

Table 10 – Get Verifiable Credential

FAME TECHNICAL INTERFACE	
Technical Interface ID	#FAAID-VII-1
Endpoint Name	issuer credential
Endpoint Description	This APIs is used to issue credential to user
Component	Issuer Service
Endpoint URI	/credential-offers
HTTP Method	POST
Request (Query) Parameters	N/A
Request Body (example)	{"credentials":["FAMEv1"],"grants":{"urn:ietf:params:oauth:grant-type:pre-authorized_code":{"pre-authorized_code":"u1yQKqJkwf7WanYr5qfsjN","user_pin_required":false}},"credentialDataSupplierInput":{"uid":"fame_09tc5t429","affiliation":"FAME","region":"Ireland","role":"admin","nickname":"John"}}
Request Headers	N/A
Response Body	{ "uri":"openid-credential-offer://?credential_offer=%7B%22grants%22%3A%7B%22urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Apre-authorized_code%22%3A%7B%22pre-authorized_code%22%3A%22u1yQKqJkwf7WanYr5qfsjN%22%2C%22user_pin_required%22%3Afalse%7D%7D%2C%22credentials%22%3A%5B%22FAMEv1%22%5D%2C%22credential_issuer%22%3A%22https%3A%2F%2Fidentity.fame-horizon.eu%22%7D", "userPinRequired": false}
Response Status Code	200 - OK

Table 11 – check credential status.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#FAAID-VII-2
Endpoint Name	Credential-offer-status
Endpoint Description	This APIs is used to check the credential status
Component	Issuer Service
Endpoint URI	/credential-offers-status
HTTP Method	POST
Request (Query) Parameters	N/A

Request Body (example)	{"id":"kdrkAdKEDLuMCuh8gy3qmm"}
Request Headers	N/A
Response Body	{"createdAt":1732538719328,"lastUpdatedAt":1732538719328,"status":"OFFER_CREATED"} {"createdAt":1732538719328,"lastUpdatedAt":1732538942251,"status":"CREDENTIAL_ISSUED"}
Response Status Code	200 - OK

Table 12 – auth request.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#FAAID-VII-3
Endpoint Name	Auth-requests
Endpoint Description	This APIs is used to verify credential from a user
Component	Verifier-Service
Endpoint URI	/auth-requests
HTTP Method	POST
Request (Query) Parameters	N/A
Request Body (example)	NA
Request Headers	N/A

Response Body	{
	
	"correlationId": "09023416-5416-4d07-9e23-c40909559c88",
	
	"definitionId": "FAMEv1",
	
	"authRequestURI": "openid://?request_uri=https%3A%2F%2Fidentity.fame-horizon.eu%2Fsiop%2Fdefinitions%2FFAMEv1%2Fauth-requests%2F09023416-5416-4d07-9e23-c40909559c88",
	
	"authStatusURI": "https://identity.fame-horizon.eu/webapp/auth-status"
	
	}

Response Status Code	200 - OK

Table 13 – auth status.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#FAAID-VII-4
Endpoint Name	Auth-status
Endpoint Description	This APIs is used to issue credential to user
Component	verifier Service
Endpoint URI	/auth-status
HTTP Method	POST
Request (Query) Parameters	N/A
Request Body (example)	{"correlationId":"09023416-5416-4d07-9e23-c40909559c88","definitionId":"FAMEv1"}
Request Headers	N/A

Response Body	{
	"status": "created",
	"correlationId": "09023416-5416-4d07-9e23-c40909559c88",
	"definitionId": "FAMEv1",
	"lastUpdated": 1732539444939
	}
	{
	"status": "sent",
	"correlationId": "09023416-5416-4d07-9e23-c40909559c88",
	"definitionId": "FAMEv1",
	"lastUpdated": 1732539732544
	}

Response Status Code	200 - OK

3.1.6 Mobile app UI

This section contains some screenshots from the FAME Identity Wallet mobile app. The figure 8 shows the user's mobile device screen when the app is launched (left), and when the user is asked by the app to unlock the encrypted local wallet (right), where the Verifiable Credentials of the user are securely stored. The encryption PIN is defined by the user when the app is installed on their device. The PIN must then be provided by the user each time the app is launched and on every execution of a privacy-sensitive operation – e.g., presenting a Verifiable Credential to a third party for verification (see next paragraph).

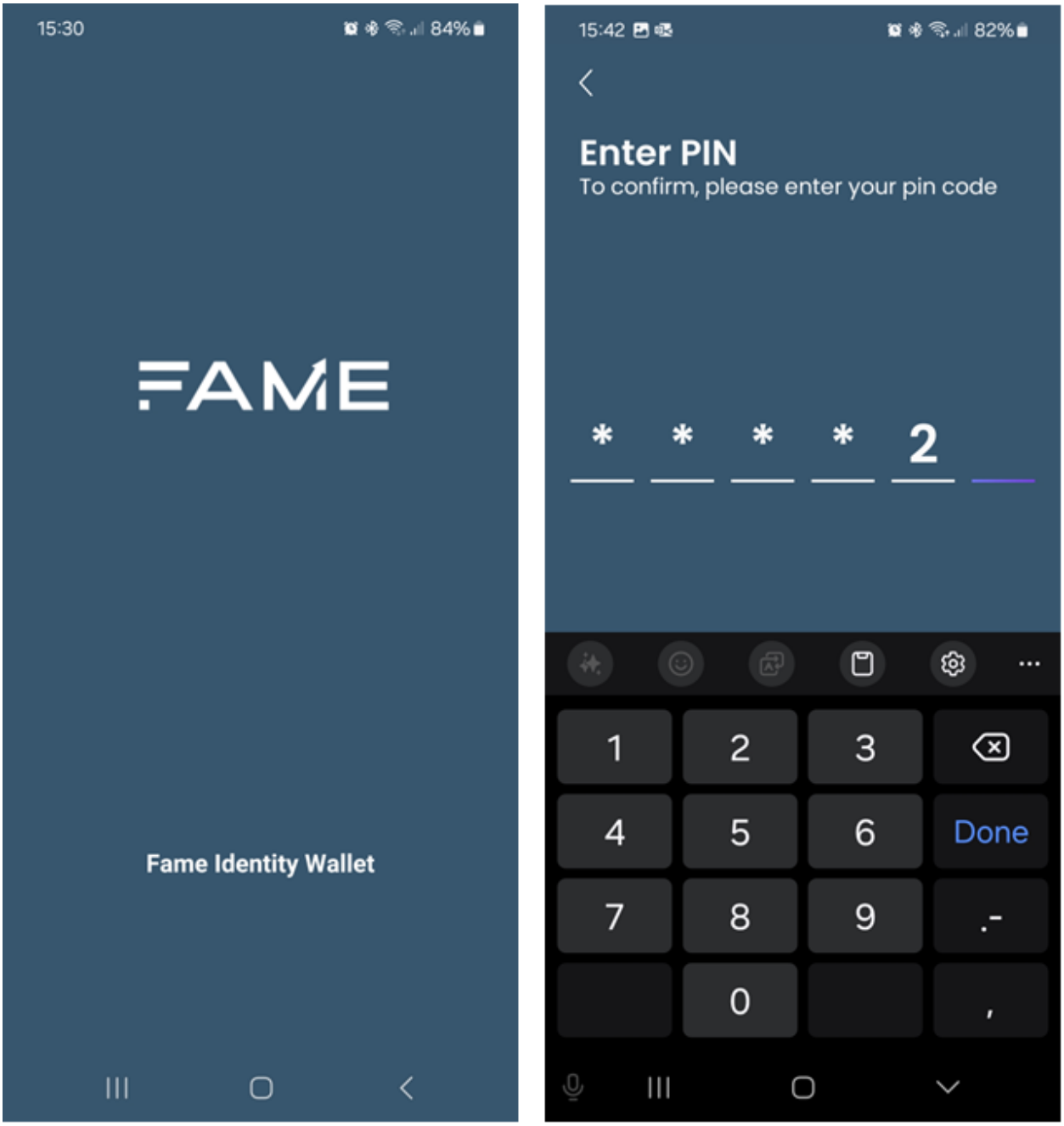

Figure 8 – FAME Identity Wallet: splash and unlock screens.

Note: The FAME ID Wallet mobile app final look and feel may differ from what presented here, more functionalities and fine tuning appearance may occur in the final period of the FAME project.

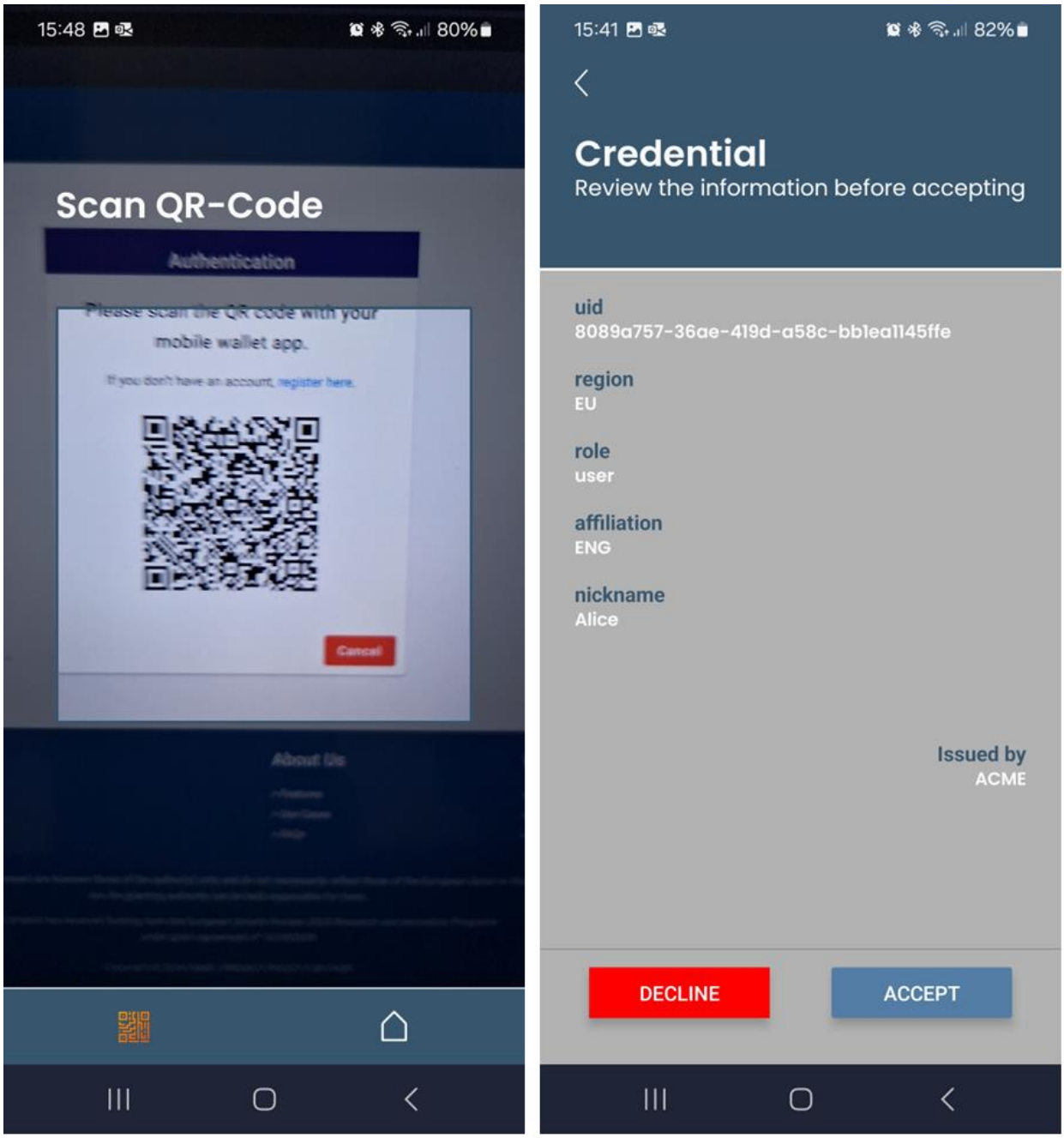

Figure 9 – FAME Identity Wallet: QR-Code scan and VOC presentation screens

The figure 9 above shows the user's mobile device screen when the user is asked to authenticate their identity (left), and when the user is asked by the app to confirm their intention to present a FAME VOC as a proof of identity during the FAME Dashboard login process.

3.2 Integrated Assets Policy Manager

3.2.1 Description

The Integrated Assets Policy Manager consists of two distinct modules, each playing its role in ensuring secure and efficient asset management within FAME. The first module, the Assets Policy Manager (APM) (figure 10), oversees the lifecycle of policies for federated assets, determining who is authorized to view or acquire them. This module ensures secure and controlled access to assets, contributing to the overall efficiency of asset management within FAME. The second module, the Dynamic Policy Manager, is an experimental component that adapts access policies based on real-time conditions. While its full potential is still being explored, it demonstrates the ability to manage dynamic factors, such as energy consumption for AI/ML microservices in WP5, and trigger policy adjustments on the fly.

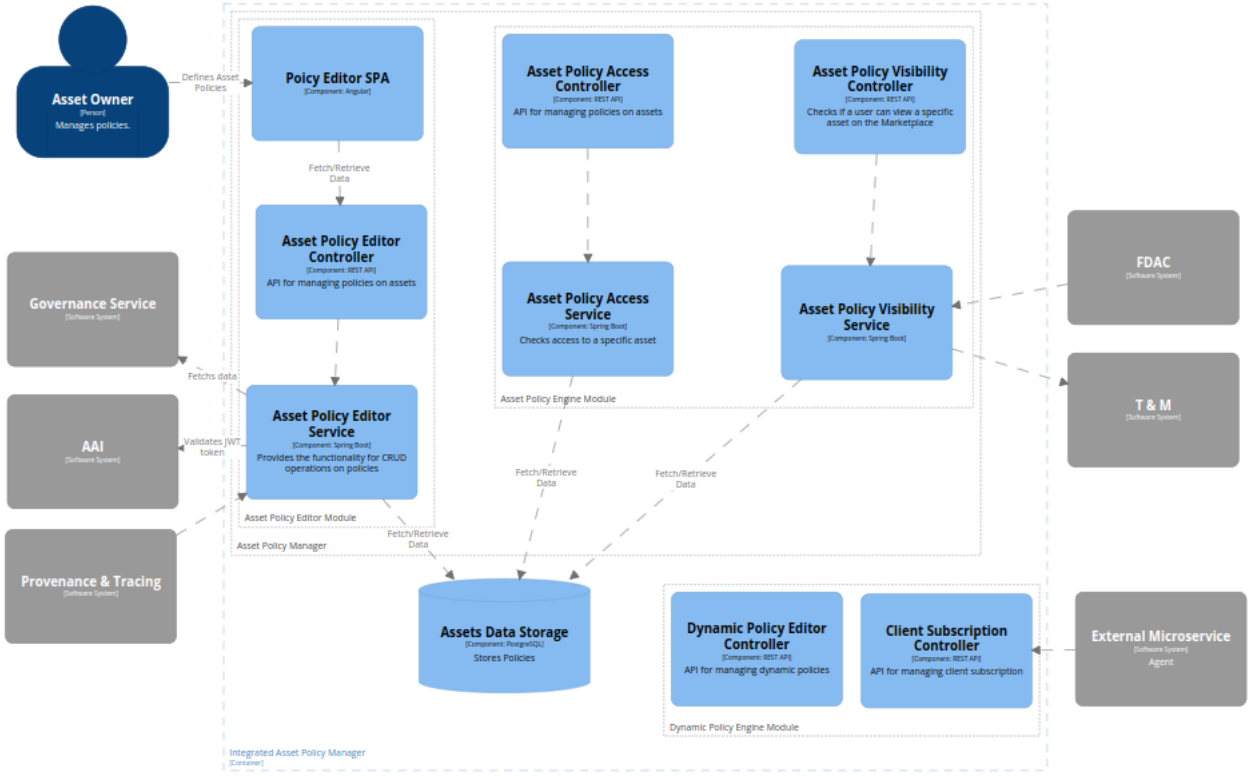

Figure 10 – 3rd level of the C4 diagram of the Integrated APM Component

3.2.2 Assets Policy Manager (APM)

3.2.2.1 Description

The Assets Policy Manager (APM) plays a crucial role in ensuring the secure management of assets within FAME. This component serves two key functionalities, both of which are vital for the smooth operation of FAME.

Firstly, the Assets Policy Manager enables the complete lifecycle management of policies associated with the federated assets discoverable through FAME, determining who is eligible to view and potentially acquire specific assets within FAME. Leveraging the Rule-Based Access Control (RBAC)

model, the component considers various user and organizational attributes to combine them in Boolean expression formatted rules to make informed policy decisions. By fulfilling its role as a Policy Decision Point (PDP), the Assets Policy Manager ensures that other components of the project always display the appropriate assets to authenticated and authorized individuals or organizations. It acts as a central authority, facilitating the enforcement of access controls throughout the system.

Secondly, the component provides end-users with a comprehensive list of the assets that they have access to their contents. This encompasses assets uploaded by the end-users themselves or any other member of their organization they are affiliated with, as well as assets acquired by them and have active contracts. This functionality offers the end-users a clear and consolidated view of their assets, enhancing their ability to manage and track their assets portfolio effectively. This functionality enhances transparency and accountability within the project, allowing users to have a clear view of the assets under their control.

Overall, the APM (figure 11) plays a critical role within FAME, since it serves as the central authority for managing asset security policies and ensuring that the appropriate assets are displayed throughout the system. By acting as a PDP, it enables effective access control enforcement by other components. Additionally, the component provides the end-users with a comprehensive overview of their owned assets, enabling better asset management and control. With its pivotal functionalities, the APM reinforces the project's security posture, fosters transparency, and promotes efficient assets management practices.

3.2.2.2 Related Work

The APM has been developed in the context of the FAME project from the ground-up, capitalizing upon open-source technologies and tools, towards resolving a series of technical requirements as these have been documented in the FAME deliverable D2.1 (Requirements, Specifications and Co-Creation), and more specifically towards addressing:

- TR003: Be able to manage and enforce access and visibility restrictions on my assets, based on defined criteria (including e.g., organization type, user role, locality etc.). (also applicable to TR110, TR207, TR607, TR715).
- TR007: Be able to define security and privacy policies.
- TR124: Be able to manage policies.
- TR203: Be able to offer of my assets only to registered user.
- TR204: Be to define the appropriate access policies on my assets.
- TR705: Be able to make my industrial data assets available through FAME under my preferred license schemes.
- TR302: Be able to implement access control within the system.
- TR406: Be able to control data privacy policies to provide a solution to any financial entity.

3.2.3 Technical Specification

3.2.3.1 Component-level C4 Architecture

The Assets Policy Manager component consists of two distinct modules: the Assets Policy Editor and the Assets Policy Engine. Each module plays a crucial role in enabling end users to define and enforce access policies for all the assets within FAME.

The first module, namely the Assets Policy Editor, empowers asset owners within FAME to define access policies that regulate who can view and potentially purchase their assets in the platform. This functionality is realized through three levels of access: Confidential, Public, and Restricted. With Confidential access, only the owner of the asset can view it, ensuring utmost privacy and exclusivity. Public access allows all the authenticated users to view the asset, promoting open sharing and collaboration. Restricted access provides more granular control, allowing asset owners to define a set of and/or conditions based on user and organizational attributes (including but not limited to country, organization, etc.), following the Rule-Based Access Control (RBAC) model. These conditions act as eligibility criteria for other users to meet for viewing the asset. The Assets Policy Editor incorporates a user-friendly UI to facilitate the intuitive definition of policies. Additionally, it provides a REST API that can be leveraged by the UI itself, as well as other components within the FAME platform, such as the Federated Data Assets Catalogue (FDAC) and Provenance & Tracing (P&T), to set policies during external asset indexing processes. Additionally, to present the organizational attributes federated by FAME to the user, the Asset Policy Editor communicates with the Governance Service through the Rest API.

The second module, namely the Assets Policy Engine, acts as the Policy Decision Point (PDP) within the platform. It is responsible for answering two fundamental questions related to asset's visibility and ownership for authenticated users. Firstly, it determines which assets a user can view within the platform, considering the defined access policies and user attributes. This ensures that users are presented with only the assets they are eligible to access. Secondly, the Assets Policy Engine provides information on the assets that the user has access to their contents. Content access includes assets that were uploaded by the user themselves or any other member of their organization (provenance) that the user is affiliated with, as well as assets that have been acquired with active contracts. To gather information about purchased assets, the Assets Policy Engine interacts with the Trading and Monetization module (T&M). Moreover, to extract the user and organization attributes that are necessary to regulate asset's visibility, the Assets Policy Manager communicates with the Authentication & Authorization Infrastructure (AAI) to retrieve the required information before making the appropriate access decisions.

It is important to highlight that comprising the PDP of FAME, any component that needs to present users with assets information must first contact the Policy Engine to retrieve access eligibility details. Subsequently, these components will act as the Policy Enforcement Points (PEP) of the platform, either allowing or denying access based on the information received from the Policy Engine. This approach ensures consistent and appropriate enforcement of access policies throughout the platform, maintaining a secure and controlled environment for data asset management and utilization.

By encompassing both the Assets Policy Editor and the Assets Policy Engine, the Assets Policy Management ensures that assets' access within FAME is controlled, secure, and aligns with the defined policies. The collaboration between these components allows for user-friendly policy definition and efficient policy enforcement, guaranteeing that users are presented with only the relevant assets that they can access based on their attributes and ownership. The overall idea of the abovementioned approach is depicted in the 3rd level of the C4 diagram (i.e., component diagram) that is presented in the section before.

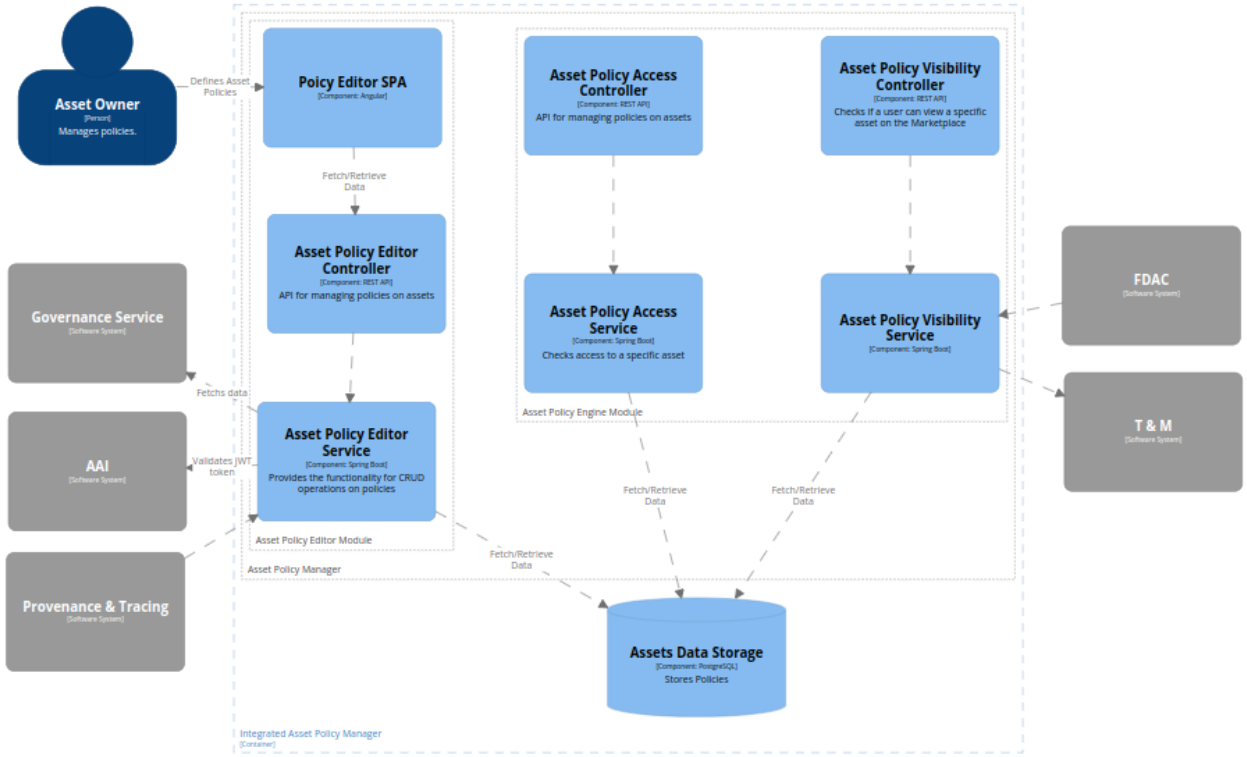

Figure 11 – 3rd level of the C4 diagram of the APM Component

As an enhancement to the Asset Policy Manager, the ability to assign specific policies at the offering level has been introduced, at the discretion of the asset owner. This enables the asset owner to control who can view and potentially purchase each specific offering of their assets. Both the Assets Policy Editor and the Assets Policy Engine interfaces have been updated to incorporate this enhancement.

3.2.3.2 Baseline Technologies and Tools

In terms of implementation, APM was built from the ground-up, capitalizing upon open-source technologies and tools. More specifically, the backend of the Assets Policy Editor and the Engine are based on Java 21 and the Spring Boot 3 framework is employed to facilitate efficient development and provide robust functionality. The data storage for the backend is handled by PostgreSQL (https://www.postgresql.org), a reliable and scalable open-source relational database management system. To enable policy definition and enforcement, both the Assets Policy Editor and the Assets Policy Engine utilize the EvalEx (https://github.com/ezylang/EvalEx) library. EvalEx is an expression evaluation library, allowing APM to efficiently define and evaluate complex rule-based expressions for the various permissions on the assets. Finally, regarding deployment, containerization solutions like Docker (https://www.docker.com) are employed to ensure consistent and reliable deployment across different environments.

3.2.3.3 Interfaces

For internal services within the same network, the X-Identity header should be included in every request when interacting with the Asset Policy Manager to ensure secure access. For external or public access, a JWT (JSON Web Token) is required, which is verified by the AAI (Authentication and Authorization Infrastructure) service. This verification ensures secure authentication and authorization, granting users access to the appropriate resources based on their permissions. The X-

Identity header will keep all the necessary information to identify the users and make decisions based on their profiles. To do so, the following JSON object should be provided as a base64 encoded string in the X-Identity header:

```
{
   "userId": "ABC",
   "organizationId": "ABC",
   "attributes": {
      "country": "Greece",
      "role": "Admin",
      "organizationName": "UBITECH",
      "organizationType": "SME",
      …,
      …
   }
}
```

The structure of the JSON contains the following info:

- userId: The unique identifier of the user
- organizationId: The unique identifier of the organization that the user belongs to
- attributes: A nested object containing additional attributes related to the user and/or the organization they belong to. This is useful to make decisions regarding what assets the users can view in the marketplace (by also combining info from the T&M module). This object can contain any other attributes that we would like to include, or even fewer than the ones stated above. However, the attributes mentioned above are a good starting point.

Table 14 – APM addPolicy Technical Interface.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#APM1
Endpoint Name	addPolicy
Endpoint Description	This endpoint is responsible for exposing the necessary functionality to create a new policy for an asset
Component	APM
Endpoint URI	/api/v1/asset-policy-editor
HTTP Method	POST
Request (Query) Parameters	N/A

Request Body (example)	{
	"assetType": "DATASET",
	"assetId": "f6f94402-fad0-4fed-8df5-60dd46831170",
	"accessType": "RESTRICTED",
	"rule": "(country == \"Greece\")"
	}

Request Headers	X-Identity: a custom header to accept the user profile in a Base64 encoded format.

Response Body	{
	"id": 1
	"assetType": "DATASET",
	"assetId": "f6f94402-fad0-4fed-8df5-60dd46831170",
	"accessType": "RESTRICTED",
	"rule": "(country == \"Greece\")"
	}

Response Status Code	201 - Created

Table 15 – APM getPolicy Technical Interface.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#APM2
Endpoint Name	getPolicy
Endpoint Description	This endpoint is responsible for fetching an existing policy
Component	APM
Endpoint URI	/api/v1/asset-policy-editor
HTTP Method	GET
Request (Query) Parameters	assetId: The ID of the asset to retrieve the policy for offeringId: (optional) The ID of the offering to retrieve the policy for
Request Body (example)	N/A
Request Headers	X-Identity: a custom header to accept the user profile in a Base64 encoded format.

Response Body	{
	"id": 1
	"assetType": "DATASET",
	"assetId": "f6f94402-fad0-4fed-8df5-60dd46831170",
	"accessType": "RESTRICTED",
	"rule": "(country == \"Greece\")"
	}

Response Status Code	200 - OK

Table 16 – APM updatePolicy Technical Interface.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#APM3
Endpoint Name	updatePolicy
Endpoint Description	This endpoint is responsible for updating an existing policy of a specific asset
Component	APM
Endpoint URI	/api/v1/asset-policy-editor
HTTP Method	PUT
Request (Query) Parameters	N/A

Request Body (example)	{
	"assetType": "FILE",
	"assetId": "f6f94402-fad0-4fed-8df5-60dd46831170",
	"accessType": "RESTRICTED",
	"rule": "(country == \"Greece\" && organizationType == \"SME\")"
	}

Request Headers	X-Identity: a custom header to accept the user profile in a Base64 encoded format.
Response Body	N/A
Response Status Code	204 – No Content

Table 17 – APM deletePolicy Technical Interface.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#APM4
Endpoint Name	deletePolicy
Endpoint Description	This endpoint is responsible for deleting a policy for an existing asset
Component	APM
Endpoint URI	/api/v1/asset-policy-editor
HTTP Method	DELETE
Request (Query) Parameters	assetId: The ID of the asset to delete the policy for offeringId: (optional) The ID of the offering to delete the policy for
Request Body (example)	N/A
Request Headers	X-Identity: a custom header to accept the user profile in a Base64 encoded format.
Response Body	N/A
Response Status Code	204 – No Content

Table 18 – APM contentAccessCheckOne Technical Interface.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#APM5
Endpoint Name	contentAccessCheckOne
Endpoint Description	This endpoint is responsible for checking if the underlying user has content access to a specific asset
Component	APM
Endpoint URI	/api/v1/asset-access/check-one
HTTP Method	GET
Request (Query) Parameters	assetId: The ID of the asset to check for content access
Request Body (example)	N/A
Request Headers	X-Identity: a custom header to accept the user profile in a Base64 encoded format.

Response Body	{
	"hasAccess": true,
	"assetAccessType": "OWN"
	}
	
	{
	"hasAccess": true,
	"assetAccessType": "BOUGHT"
	}
	
	{
	"hasAccess": false
	}

Response Status Code	200 - OK

Table 19 – APM contentAccessCheckMany Technical Interface.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#APM6
Endpoint Name	contentAccessCheckMany
Endpoint Description	This endpoint is responsible for checking if the underlying user has content access to a list of assets.
Component	APM
Endpoint URI	/api/v1/asset-policy-editor/check-many
HTTP Method	GET
Request (Query) Parameters	assetId: The ID of the asset to delete the policy for
Request Body (example)	A list of asset IDs ["db12fdba-cb79-4867-8642-896bf297374d", "a09643c2-8019-4b68-9d19-85bc3c3c7807", "2790dbeb-90e9-413c-aeda-605c44320f09"]
Request Headers	X-Identity: a custom header to accept the user profile in a Base64 encoded format.

Response Body	[
	{
	"hasAccess": true,
	"assetAccessType": " OWN "
	},
	{
	"hasAccess": true,
	"assetAccessType": "OWN"
	},
	{
	"hasAccess": false
	}
	]

Response Status Code	200 – OK

Table 20 – APM contentAccessCheckAll Technical Interface.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#APM7
Endpoint Name	contentAccessCheckAll
Endpoint Description	This endpoint is responsible for retrieving a list of all the assets that the underlying user has access to their contents.
Component	APM
Endpoint URI	/api/v1/asset-access/check-all
HTTP Method	GET
Request (Query) Parameters	assetType (OPTIONAL): Limit results based on a specific asset type.
Request Body (example)	N/A

Request Headers	X-Identity: a custom header to accept the user profile in a Base64 encoded format.

Response Body	{
	"own": [
	db12fdba-cb79-4867-8642-896bf297374d,
	a09643c2-8019-4b68-9d19-85bc3c3c7807
	],
	"bought ": [
	2790dbeb-90e9-413c-aeda-605c44320f09
	]
	}

Response Status Code	200 – OK

Table 21– APM marketplaceVisibilityCheckOne Technical Interface.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#APM8
Endpoint Name	marketplaceVisibilityCheckOne
Endpoint Description	This endpoint is responsible for checking if the underlying user can view a specific asset on the Marketplace
Component	APM
Endpoint URI	/api/v1/asset-visibility/check-one
HTTP Method	GET
Request (Query) Parameters	assetId: The ID of the asset to check for marketplace visibility
Request Body (example)	N/A
Request Headers	X-Identity: a custom header to accept the user profile in a Base64 encoded format.

Response Body	{
	"hasVisibility": false
	}

Response Status Code	200 - OK

Table 22 – APM marketplaceVisibilityCheckMany Technical Interface.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#APM9
Endpoint Name	marketplaceVisibilityCheckMany
Endpoint Description	This endpoint is responsible for checking if the underlying user can view a list of assets on the marketplace
Component	APM
Endpoint URI	/api/v1/asset-visibility/check-many
HTTP Method	GET
Request (Query) Parameters	N/A
Request Body (example)	["db12fdba-cb79-4867-8642-896bf297374d", "a09643c2-8019-4b68-9d19-85bc3c3c7807"]

Request Headers	X-Identity: a custom header to accept the user profile in a Base64 encoded format.

Response Body	[
	{
	"hasVisibility": false
	},
	{
	"hasVisbility": true
	}
	]

Response Status Code	200 - OK

Table 23 – APM marketplaceVisibilityCheckAll Technical Interface.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#APM10
Endpoint Name	marketplaceVisibilityCheckAll
Endpoint Description	This endpoint is responsible for retrieving a list of all the asset IDs that the underlying user can view on the marketplace
Component	APM
Endpoint URI	/api/v1/asset-visibility/check-all
HTTP Method	GET
Request (Query) Parameters	assetType (OPTIONAL): Limit results based on a specific asset type.
Request Body (example)	N/A
Request Headers	X-Identity: a custom header to accept the user profile in a Base64 encoded format.
Response Body	["db12fdba-cb79-4867-8642-896bf297374d", "a09643c2-8019-4b68-9d19-85bc3c3c7807"]
Response Status Code	200 - OK

Table 24 – APM marketplaceVisibilityCheckOffering Technical Interface.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#APM11
Endpoint Name	marketplaceVisibilityCheckOffering
Endpoint Description	This endpoint is responsible for checking if the underlying user can view a specific offering on the Marketplace
Component	APM
Endpoint URI	/api/v1/offering-visibility/check-one
HTTP Method	GET
Request (Query) Parameters	offeringId: The ID of the asset to check for marketplace visibility
Request Body (example)	N/A
Request Headers	X-Identity: a custom header to accept the user profile in a Base64 encoded format.

Response Body	{
	"hasVisibility": false
	}

Response Status Code	200 - OK

Table 25 – APM marketplaceVisibilityOfferingRetrieveAll Technical Interface.

FAME TECHNICAL INTERFACE	
Technical Interface ID	#APM10
Endpoint Name	marketplaceVisibilityOfferingRetrieveAll
Endpoint Description	This endpoint is responsible for retrieving a list of all the offering IDs that the underlying user can view on the marketplace for a specific asset
Component	APM
Endpoint URI	/api/v1/offering-visibility/retrieve-all
HTTP Method	GET
Request (Query) Parameters	assetId: The ID of the asset to check for marketplace visibility
Request Body (example)	N/A
Request Headers	X-Identity: a custom header to accept the user profile in a Base64 encoded format.
Response Body	["off12fdba-cb79-4867-8642-896bf297374d", "offd643c2-8019-4b68-9d19-85bc3c3c7807"]
Response Status Code	200 - OK

The above endpoints are also included in detail on the OpenAPI specification format which is available at the project Gitlab at the following URL:

```
https://gitlab.infinitech-h2020.eu/fame/framework/apm/-/blob/master/OpenAPI.yaml
```

3.2.4 Dynamic Policy Manager (DPM)

3.2.4.1 Description

The Dynamic Policy Manager (DPM) is an experimental component whose purpose is to control access to microservice assets according to policies that may depend on real-world conditions. Obviously, this concept has a wide range of possible applications, but as a concrete case and a demonstration of dynamic policy capabilities we have proposed access control to manage energy costs, which is particularly relevant for the AI/ML microservices developed in WP5. It is known that AI/ML algorithms can consume significant amounts of energy, first during the training phase and then during the inference phase. The training phase is energy consuming due to factors such as the complexity of the models, the volume of data required for training, and the duration of the training phase. The inference phase is generally less energy consuming, but since the energy consumed for inferences is proportional to the number of requests, the total energy costs for this phase may also become significant when a large number of users is involved over long periods of time.

Dynamic access policies are not intended to replace traditional access policies, such as attribute-based access control or role-based access control, but rather to complement them with advanced capabilities.

3.2.5 Technical Specification

The purpose of the Dynamic Policy Manager is to enable management of dynamic policies for microservices. In this particular case, the objective of the policies is to curb the energy costs of microservices while still maintaining an acceptable service level for clients using these services. This involves making an automatic trade-off between client needs and resource costs. The access control system works by making access control decisions based on a number of static and dynamic information items. In this case the information items are the following:

- *Subscription class*. Each client is assigned a subscription class that determines its importance, and hence, its expected service level.
- *Invoked function*. This consists of the endpoint and the access method of the particular microservice requested.
- *Function arguments*. These are the arguments to the function, including the set of query parameters in the URL, and for some access methods also the document in the request body.
- *Estimated energy*. This is obtained from an Energy Estimation API to be implemented for each particular microservice subject to this type of dynamic access policy. It provides an estimate of the energy required to carry out the invoked function with the provided arguments.
- *Current cost of energy*. This is the current cost of electrical energy in the zone where the server running the microservice is located. This information is provided in the form of RESTful microservices by organizations like ENTSO-E and others.
- *Request history*. This consists of the history of successful requests made by the client and the associated energy costs.
- *Policy expression*. This is a Boolean expression that combines the above-mentioned elements in order to make an access decision for a request with respect to its energy consumption. Before this decision is made, the request may have been subject to other access decisions, but this is outside the scope of the dynamic access control system.

When a given client makes an access request that is denied due to the energy cost policy, it receives an HTTP 429 response code (Too Many Requests). In this case it may retry the request after a certain time, which may then either succeed or be denied again. The total effect is that client requests will be rate-limited based on a trade-off between the importance of the client and the cost of energy required to carry out its requests. A well-formulated policy will ensure an acceptable service level for important clients while still not starving less important clients indefinitely.

Policy expressions are written in terms of four variables, five functions, and a standard set of arithmetic and logic functions.

The variables are the following:

- s: a subscriber ID
- t: the current time since epoch [s]
- f: an API function consisting of a method (like GET, POST, PUT) and an endpoint
- a: arguments to an API function (in the URL or in the request body)

The functions are the following:

- $\mathrm{Upr}(t)$: the unit price of electrical energy at time t in the local energy zone [EUR/J]
- $\mathrm{Eng}(f, a)$: the estimated energy consumption of API function f called with arguments a [J]

- $\mathrm{Cls}(s)$: the subscription class of s
- $\mathrm{Prc}(t, f, a)$: the energy price of executing a given function, f, with a set of arguments, a, at time t; it is defined as $\mathrm{Prc}(t, f, a) = \mathrm{Upr}(t) \cdot \mathrm{Eng}(f, a)$
- $\mathrm{Hst}(s, t_s, t_e)$: the cumulative estimated energy of all granted accesses for client s in the period from t_s to t_e

In addition to these elements, the following arithmetic and logic operators are available:

- $+, -, \cdot, /, =, <, \leq, >, \geq, \neg, \wedge, \vee$

Access decisions are determined by the policy function, $p(t, s, f, a) = \alpha$, where α is a Boolean policy expression. The following is an example of a policy expression:

$$\begin{aligned}&(\mathrm{Cls}(s) = 1) \ \vee \\ &\big(\mathrm{Cls}(s) = 2 \wedge \big(\mathrm{Prc}(t, f, a) + \mathrm{Hst}(s, t - 20, t)\big) \leq 10\big) \ \vee \\ &\big(\mathrm{Cls}(s) = 3 \wedge \big(\mathrm{Prc}(t, f, a) + \mathrm{Hst}(s, t - 60, t)\big) \leq 5\big)\end{aligned}$$

The first clause of the disjunction says that requests from subscribers of class 1 are always granted. The second clause says that for subscribers of class 2 the combined energy consumption of the current API call and the granted calls over the last 20 seconds must not exceed 10 joules. The third clause says that for subscribers of class 3 the combined energy consumption of the current API call and the granted calls over the last 60 seconds must not exceed 5 joules.

When an access request is made, the four variables t, s, f, a are extracted, and then the policy function is evaluated with these as arguments. The request is denied if the function value is false. (As discussed earlier it may also be denied for reasons unrelated to the dynamic policy.)

The policy expression is encoded in JSON syntax according to the following two rules:

1. Each variable name, function, or operator is encoded as a JSON string. The following operators have special string encodings:
    a. Multiplication, ·, is encoded as `"*"`.
    b. Less than or equal to, ≤, is encoded as `"<="`.
    c. Greater than or equal to, ≥, is encoded as `">="`.
    d. Logical not, ¬, is encoded as `"not"`.
    e. Logical and, ∧, is encoded as `"and"`.
    f. Logical or, ∨, is encoded as `"or"`.
2. Each application of a function or an infix operator is encoded as an array whose first element is the name of the function or operator, and the remaining elements are the arguments.

Applying these rules to the policy expression example above, we encode this in JSON as follows:

```
["or",
 ["=", ["Cls", "s"], 1],
 ["and", ["=", ["Cls", "s"], 2],
         ["<=", ["+", ["Prc", "t", "f", "a"], ["Hst", "s", ["-", "t", 20], "t"]], 10]],
 ["and", ["=", ["Cls", "s"], 3],
         ["<=", ["+", ["Prc", "t", "f", "a"], ["Hst", "s", ["-", "t", 60], "t"]], 5]]
]
```

The following information elements are configured by DPM management functions and made available to the policy function:

- Local energy zone. Microservices offered by ENTSO-E and other organizations provide the current, past, and future prices of electrical energy. Each energy zone (or "bidding zone") has its own price. The management function needs to configure this for the location of the servers running the target microservices, so that the $\mathrm{Upr}(t)$ function can obtain prices for the correct energy zone.
- Energy estimation endpoint. This is an endpoint that mirrors the target endpoint and arguments, but instead of performing AI/ML operations, it returns an estimate of the energy that would be required to carry out these operations.
- Subscription class. Each client has a subscription class, which is also configured using DPM management functions.

The Dynamic Policy Manager is currently implemented as a proof-of-concept and has been used to perform simulations. A scientific article describing the concept and application is in progress and will be submitted to a journal.

3.2.5.1 Interfaces

The DPM supports four management interfaces: two for managing the dynamic policies themselves, and two for managing the client subscription classes.

Table 26 – DPM setPolicy Technical Interface

FAME TECHNICAL INTERFACE	
Technical Interface ID	#DPM0
Endpoint Name	dpmSetPolicy
Endpoint Description	Set the policy for a given service endpoint
Component	DPM
Endpoint URI	/api/v1/dpm/policy
HTTP Method	POST
Request (Query) Parameters	
Request Body (example)	{ "service_endpoint": "/example/endpoint", "energy_zone": "NO1", "energy_estimation_endpoint": "/energy/example/endpoint", "policy": ["or", ["and", ["=", ["Cls", "s"], 1], true], ["and", ["=", ["Cls", "s"], 2],

	["<=", ["+", ["Prc", "t", "f", "a"], ["Hst", "s", ["-", "t", 20], "t"]], 10]], ["and", ["=", ["Cls", "s"], 3], ["<=", ["+", ["Prc", "t", "f", "a"], ["Hst", "s", ["-", "t", 60], "t"]], 5]]] }
Request Headers	X-Identity: a custom header to accept the user profile in a Base64 encoded format.
Response Body	
Response Status Code	201 – Created 400 – Bad Request 401 – Unauthorized

Table 27 – PDM getPolicy Technical Interface

FAME TECHNICAL INTERFACE	
Technical Interface ID	#DPM1
Endpoint Name	dpmGetPolicy
Endpoint Description	Get the policy for a given service endpoint
Component	DPM
Endpoint URI	/api/v1/dpm/policy
HTTP Method	GET
Request (Query) Parameters	service_endpoint=%2Fexample%2Fservice
Request Body (example)	N/A
Request Headers	X-Identity: a custom header to accept the user profile in a Base64 encoded format.
Response Body	{ "service_endpoint": "/example/endpoint", "energy_zone": "NO1", "energy_estimation_endpoint": "/energy/example/endpoint", "policy": ["or", ["and", ["=", ["Cls", "s"], 1], true], ["and", ["=", ["Cls", "s"], 2],

	["<=", ["+", ["Prc", "t", "f", "a"], ["Hst", "s", ["-", "t", 20], "t"]], 10]], ["and", ["=", ["Cls", "s"], 3], ["<=", ["+", ["Prc", "t", "f", "a"], ["Hst", "s", ["-", "t", 60], "t"]], 5]]] }
Response Status Code	200 – OK 404 – Not Found

Table 28 – DPM setClientClass Technical Interface

FAME TECHNICAL INTERFACE	
Technical Interface ID	#DPM2
Endpoint Name	dpmSetClientClass
Endpoint Description	Set the subscription class for a given client
Component	DPM
Endpoint URI	/api/v1/dpm/client/{clientID}
HTTP Method	POST
Request (Query) Parameters	
Request Body (example)	{ "class": 1 }
Request Headers	X-Identity: a custom header to accept the user profile in a Base64 encoded format.
Response Body	
Response Status Code	200 – OK

Table 29 – DPM getClientClass Technical Interface

FAME TECHNICAL INTERFACE	
Technical Interface ID	#DPM3
Endpoint Name	dpmGetClientClass
Endpoint Description	Get the subscription class for a given client
Component	DPM
Endpoint URI	/api/v1/dpm/client/{clientID}
HTTP Method	GET
Request (Query) Parameters	
Request Body (example)	
Request Headers	X-Identity: a custom header to accept the user profile in a Base64 encoded format.
Response Body	{ "class": 1 }
Response Status Code	200 – OK 404 – Not Found

4 Components Demonstration

4.1 Federated Authentication and Authorization Infrastructure (FAAID)

The AAI services manage secured credentials, allowing through APIs to issue, verify, revoke verifiable credentials in the W3C standard. Since authentication and authorization takes place through the selective detection of user attributes (i.e., W3C verifiable credentials[3], a service is therefore necessary that allows the issuance, revocation and verification.

This service acts as an "issuer" role in the self-sovereign identity model. Read more at W3C online Recommendation here: https://www.w3.org/TR/vc-data-model/#ecosystem-overview

Unlike traditional OIDC (OpenID Connect) providers that store user data in a centralized database, FAME's user management adheres to the self-sovereign identity model. In this approach, users retain exclusive control of their information within wallet applications. Authentication and authorization processes are conducted through the selective disclosure of specific user attributes.

4.1.1 Prerequisites and Installation Environment

The AAI service is available in the form of a Docker image, can be deployed in any environment (local, cloud) and in any operating systems (Windows, Mac OS, Linux), if Docker and docker-compose plugin are installed. The Docker image is available on project's Harbor, which is container registry at the following URL: https://harbor.gftinnovation.eu/fame/fame-aai-identity-ssi-agent.

In addition to Docker Compose deployment scripts, Kubernetes deployment capabilities are also supported, enabling seamless automation of continuous integration and deployment through tools like Harbor [Harbor], Argo CD [Argo-CD], and GitLab CI [GITLAB]. Kubernetes orchestrates containerized applications, offering advanced features such as automated scaling, self-healing, and efficient management across diverse environments. By integrating Argo CD into the Kubernetes workflow, deployments are managed declaratively, ensuring consistency and reliability throughout development, and production stages. This approach not only accelerates the release cycle but also enhances rollback functionality, monitoring, and security. Kubernetes configuration is available on the following URL under AAI folder https://gitlab.gftinnovation.eu/fame/fame-integration.git.

4.1.2 Installation Guide

To deploy aai-identity-ssi service in any environment, you can utilize the provided docker-compose.yaml file which is available on the following URL: https://gitlab.gftinnovation.eu/fame/framework/aai.git

[3] https://www.w3.org/TR/vc-data-model/#what-is-a-verifiable-credential)

Ensure the following prerequisites are met before proceeding:

- The required file (e.g, docker-compose.yml) is available in the working directory.
- Both Docker and Docker Compose are installed on your system.

Steps to Start the Service:

- Open a terminal or command prompt.
- Navigate to the directory containing the docker-compose.yml file.
- Run the following command to start the AAI Identity SSI service:
  docker-compose up -d
- The service will start, assuming port 8026 is available for use.

4.1.3 Local development using docker

To start-ups the project in development way using docker you must run the following command in the project root. The first time it will take a while (be patience), since it must build images and download all the npm dependencies.

```
docker-compose -f docker/compose/build/docker-compose.yml build
docker-compose -f docker/compose/build/docker-compose.yml up -d
```

You can stop the container at any time, just run the below command.

```
docker-compose -f docker/compose/build/docker-compose.yml down
```

4.1.4 User Guide

Here's a step-by-step guide on how to complete the issue VOC credential based on the following two pages.

Page 1: Manually fill out details

a. On first page, you are prompted to manually fill out data to claim your verifiable credential (figure 12)
b. Fill the following fields:
   - Affiliation:
   - Region
   - Role
   - Nick name
c. Once you've entered all the required information, click the "Share" button to proceed.

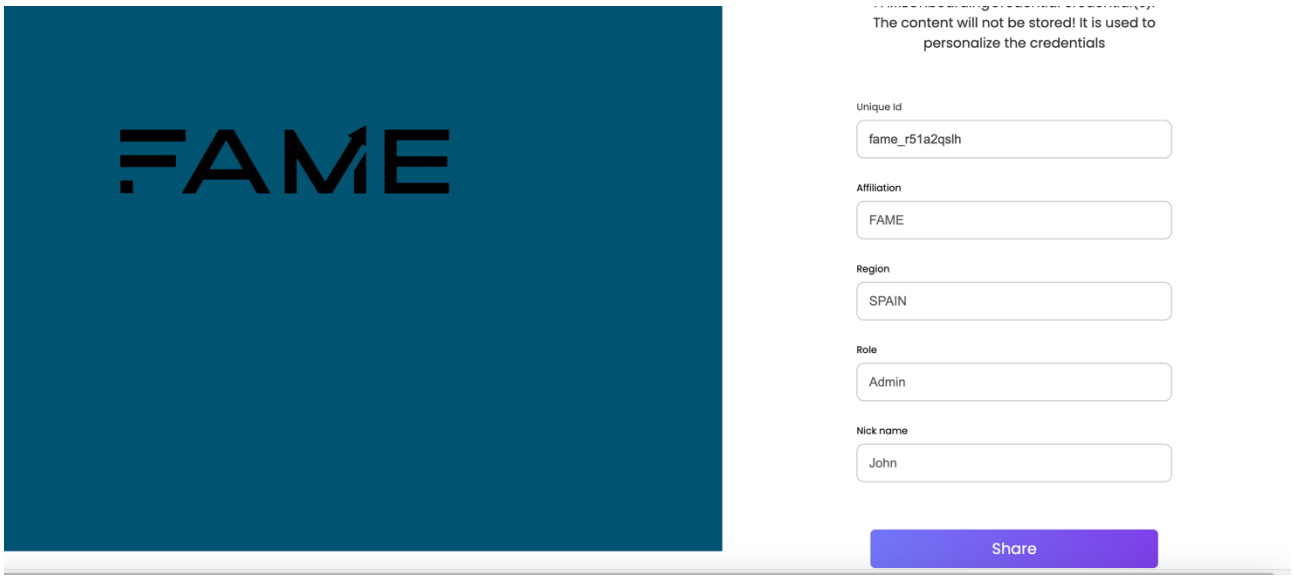

Figure 12 – Verifiable Credential issuer page

Page 2: Secure your credential.

a. On the second page, you are shown a QR code (figure 13).
b. Use your FAME identity wallet's QR code scanner to accept and store the verifiable Credential.
c. After scanning the QR code with your FAME Identity mobile wallet app, your credential will be securely added to your wallet.

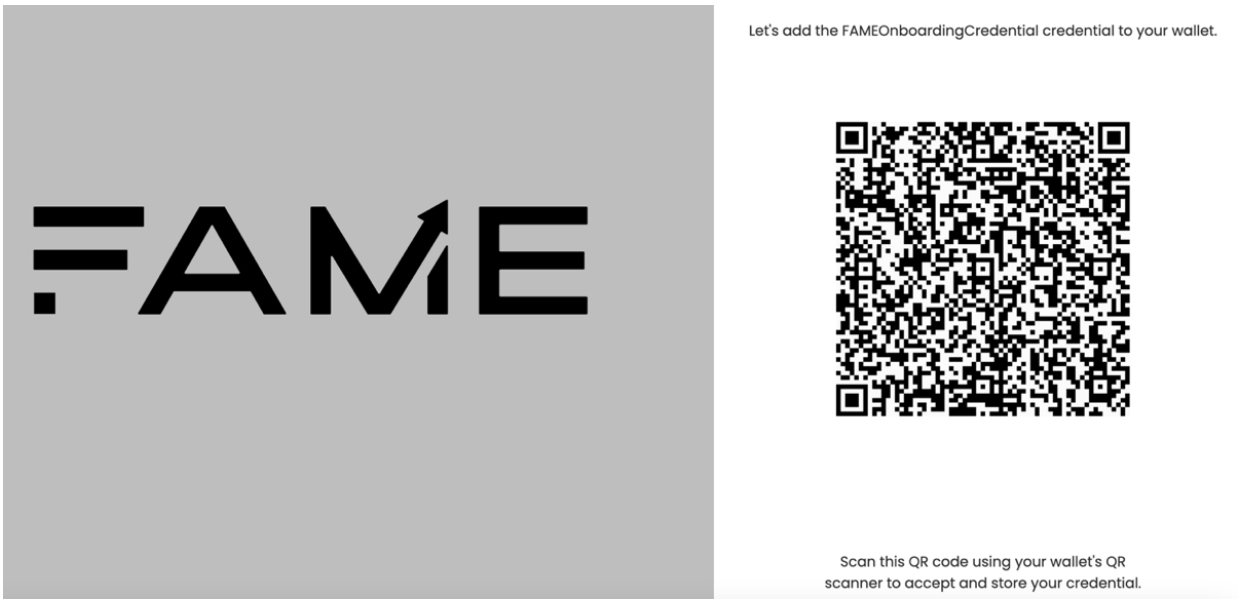

Figure 13 – Scan QR code page

On FAME identity wallet:

d. Launch the wallet app
e. Navigate to the QR reader at the bottom left (figure 14).
f. Scan the QR codes as shown in above screen.

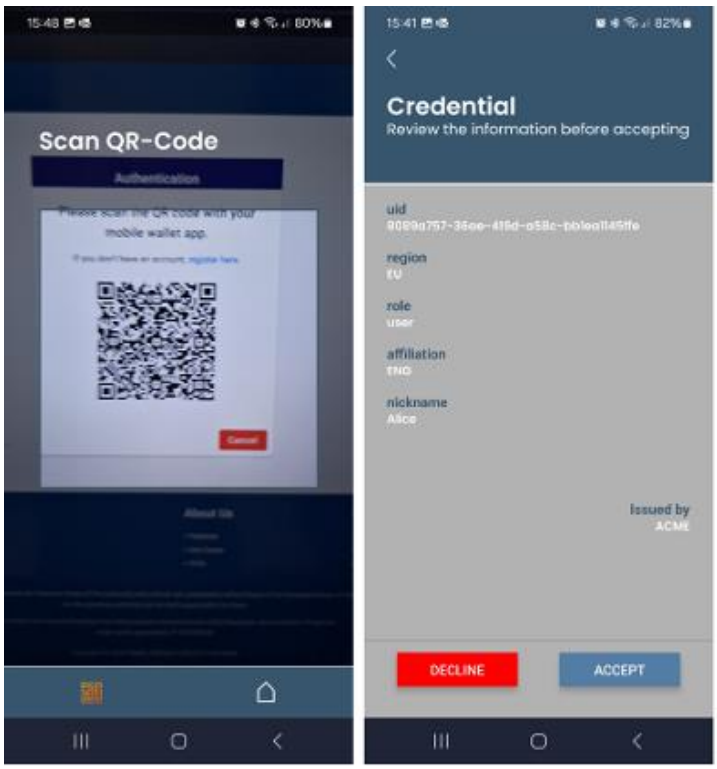

Figure 14 – scan QR and credential detail

You will now be directed to the Credential Offer screen, which displays the offered credential. Review the Credential Offer and choose to either accept or decline it as per the above figure.

By following these steps, you have successfully claimed and secured your verifiable credential.

To complete the Login/authentication process, please follow these steps:

1. Use your mobile Fame Identity wallet app to scan the QR code displayed on the screen (figure 15).
2. If you don't have a VOC credential in Fame identity wallet app yet, you can register for one as detailed in previous steps.

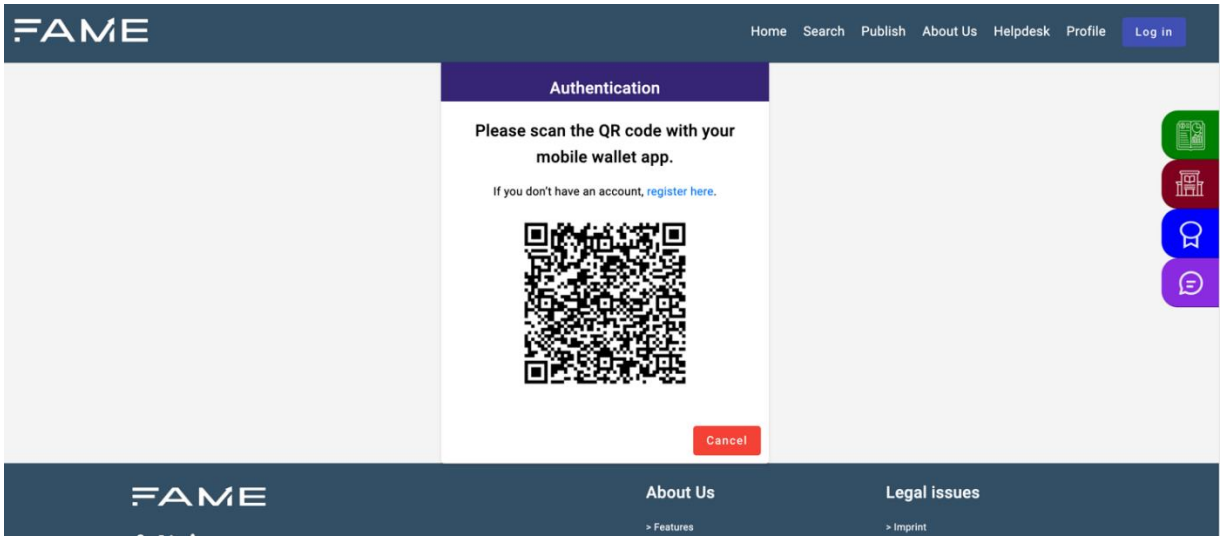

Figure 15 – Login page with QR code

1. Launch the wallet.
2. Navigate to the QR reader at the bottom left.
3. Scan the QR codes as shown in above screen, and
4. You will now be directed to the overview screen, where you can select the required Verifiable Credentials from your wallet.
5. On the Available Credentials screen, you can directly tap/select the desired credential. Once selected, the Share button will be enabled, and a green checkmark will appear next to the input requirement. Tap the Share button to proceed.
6. The credential has been successfully shared with the Verifier, and the Verifier's system will confirm your login, updating the login button to display as 'Logout'.

4.2 Federated Assets Policy Manager (FAPM)

4.2.1 Prerequisites and Installation Environment

The Assets Policy Manager is available in the form of a Docker image. APM can be deployed in any environment (local, cloud) and in any operating system (Windows, Mac OS, Linux), if Docker and docker-compose plugin are installed. The Docker image is available on project's Harbor, which is a container registry, at the following URL:

```
https://harbor.infinitech-h2020.eu/harbor/projects/40/repositories/framework%2Fapm
```

Except from using Docker Compose deployment scripts, Kubernetes deployment capabilities are also supported, automating continuous integration and deployment through tools like Harbor [Harbor], Argo CD [Argo-CD] and GitLab [GITLAB] CI. Kubernetes orchestrates containerized applications, enabling features like automated scaling, self-healing, and efficient management across different environments. With Argo CD integrated into the Kubernetes workflow, deployments are handled in a declarative manner, ensuring consistency and reliability throughout development, staging, and production. This approach accelerates the release cycle while maintaining robust rollback, monitoring, and security features. Kubernetes configuration is available on the following URL under APM folder:

```
https://gitlab.gftinnovation.eu/fame/fame-integration.git
```

4.2.2 Installation Guide

To deploy APM in any environment, you can utilize the provided docker-compose.yaml file which is available on the projects Gitlab at the following URL:

```
https://gitlab.gftinnovation.eu/fame/framework/apm
```

As long as this file is available and Docker alongside with docker-compose are installed, you can run the **docker-compose up –d** command to start up the APM. The specific installation assumes that ports 5432 and 8080 are not being already used by another process. If any of those are already occupied, one should first modify the exposed ports on the compose file. Furthermore, the default image of APM that is used is a GraalVM Native compiled one. Native images provide significant performance boost and resource utilization compared to typical images. However, some hardware architectures do not support native images and it might be the case that during the component start-up, the process will fail. If such a problem is encountered, modify the image name in the compose file and switch from the Native image to the plain one, since both image versions are available.

4.2.3 User Guide

A detailed description of the use of the APM component is available in the form of a reference video, which is currently hosted in the project repository and is available to the project partners. Once the FAME Learning Hub matures, the reference video will also be made available through the FAME Learning Hub.

4.3 Dynamic Policy Manager (DPM)

A proof of concept of the DPM has been developed, enabling simulations of dynamic policies and access decisions. Figure 16 shows a simulation of three clients, one from each subscriber class, performing the same function call with the same parameters at five-second intervals under the access policy expression example given in Section 0. In this simulation, the unit price of energy starts out at 0.5, then rises to 1.0 at the 40-second mark, and then falls to 0.1 at the 90-second mark. As the policy specifies, all access requests made by the client in class 1 are granted. For the client in class 2, some access requests are granted, and some are denied. Even after the unit price of energy is doubled, it retains some level of service. When the unit price of energy drops to a tenth of this level, all access requests are granted. The client in class 3 gets only minimal service when the unit price of energy is 0.5. When the price doubles, it gets no service at all. When the unit price of energy drops to a tenth, it gets a higher level of service, but there are still some accesses that are denied.

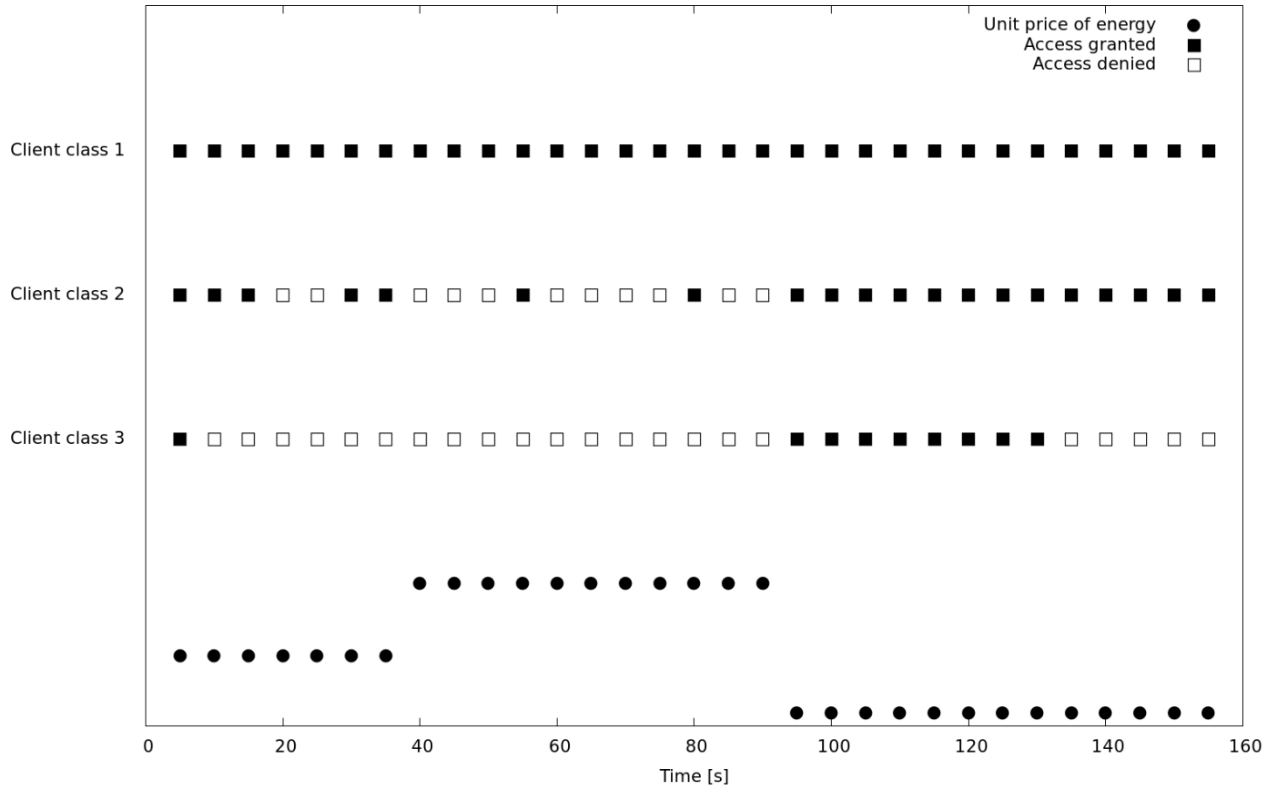

Figure 16 – Simulation of dynamic policy

5. Conclusions

The deliverable "D3.4 – Secure Federated Data Management II" reports the effort of tasks 3.1 and 3.4 to build a comprehensive solution that provides a secure and user-centric federated management framework. The deliverable has been built upon the foundational work of D3.1, highlighting the implementation, functionality, architecture, and user engagement updates to support seamless and secure access to federated assets. The integration of (FAAID), (FAPM), and the development of the experimental (DPM) tool ensures comprehensive control over authentication, access, and policy enforcement. Together, these components provide a streamlined, scalable, and highly secure federated environment.

FAAID employs cutting-edge standards, using OpenID Connect (OIDC) and more specifically OID4VP, OID4VCI, and Siopv2 to deliver a secure, interoperable digital identity verification system. The FAAID reduces reliance on centralized identity systems and, by supporting advanced user authentication workflows through the FAME Identity Wallet, empowers users with enhanced control over their digital identities while mitigating security risks.

The Federated Assets Policy Manager builds upon an authentication mechanism to implement concrete access control and policy enforcement. By implementing a Rule-Based Access Control (RBAC) model, it ensures the management of assets within FAME. The FAPM by fulfilling its role as Policy Decision Point (PDP), provided transparency and accountability within the project by guaranteeing that only the appropriate assets are displayed to authenticated and authorized individuals and organizations.

The Dynamic Policy Manager, complementing the Assets Policy Manager with dynamic capabilities, is a cutting-edge research innovation that enables real-time dynamic control of resources based on various external conditions. The selected domain, energy cost, provides a simple yet compelling demonstration of the capabilities while also addressing relevant concerns surrounding energy consumption and carbon footprint due to the increasing adoption of AI/ML algorithms in computer systems. An expected secondary effect is that over time the increased transparency will motivate developers to develop more energy-efficient algorithms of this type.

The achievements outlined in this deliverable demonstrate enhanced security measures, streamlined user authentication processes, and efficient management of assets and policies. The FAME framework also supports regulatory compliance and ensures transparency by offering users a consolidated view of their accessible assets. Designed for scalability and interoperability, the framework is well-positioned to adapt to future advancements and broader application scenarios. As a continuation of D3.1, this deliverable showcases significant progress and provides a solid foundation for further developments in secure federated data management.

The FAME consortium has developed two versions of the FAAID framework, as outlined as follow:

Version I, detailed in D3.1, is developed following Free Open Source Software (FOSS) design principles and includes a desktop wallet to show all the capabilities and functionalities and make version I fully functional.

Version II, outlined in D3.4, focuses on FAME integration including particular requirements and addressing the latest FAME developments and functionalities required as specified for the FAME Solution Architecture.

The table 30 summarises the features and functionalities of the FAAID framework developed in Fame and the evolution between V1 and V2.

Table 30 – comparison different AAI versions

i3-MARKET AAI	
Security Design Principles	Yes
FAIR design Principles	Yes
W3C VC	Yes
DID	Yes
SSI	Yes
DLT	Yes
Federated Stack	No
User-Centric Authentication	Yes
Service-Centric	Yes
Software Wallet	Yes
Hardware Wallet	No
Open-Source	Yes
OpenID4VCI	No
OpenID4VP & SIOPv2	No
QR Code	No

FAAID V1	
Security Design Principles	Yes
FAIR design Principles	Yes
W3C VC	Yes
DID	Yes
SSI	Yes
DLT	Yes
Federated Stack	No
User-Centric Authentication	Yes
Service-Centric	Yes
Software Wallet	Yes
Hardware Wallet	No
Open Source	Yes
OpenID4VCI	No
OpenID4VP & SIOPv2	No
QR Code	No

FAAID V2	
Security Design Principles	Yes
FAIR design Principles	Yes
W3C VC	Yes
DID	Yes
SSI	Yes
DLT	Yes
Federated Stack	Yes
User-Centric Authentication	Yes
Service-Centric	Yes
Software Wallet	Yes
Hardware Wallet	Yes
Open-Source	No
OpenID4VCI	Yes
OpenID4VP & SIOPv2	Yes
QR Code	Yes
SD-JWT capability (Selective Disclosure for JWTs)	Yes

Note: The FAME consortium developed FAAID framework taking as reference the i3-MARKET project results, the FAME stakeholders requirements, and the latest developments of the Veramo [Veramo] framework, making it FAAID more secure and providing tamper-proof credentials.

The FAME consortium has defined a series of KPI's that are associated with the developed Federated Authentication and Authorization Infrastructure (FAAID) framework and the Asset Policy Manager (APM), as outlined as follow:

The Federated Authentication and Authorization Infrastructure framework implements decentralized identities by adopting Self-Sovereign Identity Standards and ensures the employment of Verifiable Credentials (VCs), and Decentralized Identifiers (DIDs).

The Asset Policy Manager supports multiple assets attributes and can provide abstracted rules that can be applied to individual assets. Theoretically the number of policy abstractions that can be applied are (n) where n is the number of attributes characterizing an asset. In the current implementation of the APM n=4 hence the total number of abstracted policies is 24.

The evaluation of policy violations was performed with all publicly available assets, which were taken from external marketplaces that have been consolidated into FAME marketplace. Therefore no policy violations could be executed. Nevertheless by modifying the attributes of the assets and querying the assets through the FDAC no violations where observed.

Table 31 – Related KPIs.

Obj. ID	**KPI ID**	**Measured KPI**	**Expected Value [*M9*]**	**Achieved Value [*M9*]**	**Expected Value [*M18*]**	**Achieved Value [*M18*]**
O2	2.1	Data Marketplaces/Spaces implementing the Data Provider Interface	0	0	3	3
O2	2.2	Security and Data Protection Policies to be abstracted	6	11	13	24
O2	2.3	Increased Prevention of Policy Violations on Data Assets from other marketplaces	13%	100%	25%	100%

6. References

1. [W3C-DIDs] W3C Credentials Community Group. (n.d.), "Decentralized Identifiers (DIDs)-W3C Working Draft.," [Online]. Available: https://w3c-ccg.github.io/did-spec.
2. [WWW-VCWG] World Wide Web Consortium, “Verifiable Claims Working Group”, 2017. [Online]. Available: https://www.w3.org/2017/vc.
3. [A. Mavrogiorgou, et al.], "FAME: Federated Decentralized Trusted Data Marketplace for Embedded Finance", [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10215814.
4. [FAME-D2.1] "D2.1 - Requirements Analysis, Specifications and Co-Creation I," 2023.
5. [OpenID-OID4VCI], "OpenID for Verifiable Credential Issuance (OID4VCI)," [Online]. Available [Online]. Available: https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html
6. [OpenID] Foundation (n.d.), "OpenID for Verifiable Presentations," [Online]. Available: https://openid.net/specs/openid-4-verifiable-presentations-1_0.html/.
7. [IBM-PII], “What is personally identifiable information (PII)?”, [Online] Available: https://www.ibm.com/topics/pii/.
8. [Sphereon-SSI-VC], “Open source SDK to build SSI and Verifiable Credential applic, “Open source SDK to build SSI and Verifiable Credential applic, “Open source SDK to build SSI and Verifiable Credential applications”, [Online]. Available: https://sphereon.com/sphereon-ssi-sdk/.
9. [EU-DID] European Commission, “European Digital Identity”, [Online]. Available: https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/europe-fit-digital-age/european-digital-identity_en.
10. [IETF-RFC7519] Internet Engineering Task Force, “RFC 7519 – JSON Web Token (JWT)”, [Online] Available: https://datatracker.ietf.org/doc/html/rfc7519.
11. [i3-MARKET], [Online]. Available: https://www.i3-market.eu/i3-market-architecture/.
12. [Kubernetes], [Online]. Available: https://kubernetes.io/.
13. [Harbor], “Harbor is an open-source registry that secures artifacts, [online]. Available: https://goharbor.io/.
14. [Argo-CD], “Argo CD is a declarative, GitOps continuous delivery tool for Kubernetes”, [Online]. Available: https://argo-cd.readthedocs.io/en/stable/.
15. [GITLAB], “Gitlab is Continuous Integration and Delivery Tool”, [Online] Available: https://about.gitlab.com/solutions/continuous-integration/.
16. [Veramo], “Veramo, a decentralized identity and credential management library for JavaScript/TypeScript”, [Online]. Available: https://veramo.io/.
17. [N. Sakimura, J. Bradley and N. Agarwal 2015], “Proof Key for Code Exchange by OAuth Public Clients”, September 2015. [Online]. Available: https://datatracker.ietf.org/doc/rfc7636/.
18. [Sphereon-Wallet], “Control your data. The Sphereon Wallet” [Online]. Available: https://sphereon.com/sphereon-products/sphereon-wallet/.
19. [React JS] React js https://react.dev/reference/react [Online]
20. [Angular JS] Angular JS https://angularjs.org/ [Online]
21. [Shah 2023] Javed Shah, what is Adaptive Authentication and Authorization? January 3, 2023, https://www.1kosmos.com/authentication/adaptive-authentication/
22. [Singh 2022] Singh, J., Patel, C., & Chaudhary, N. K. (2022, December). Resilient Risk-Based Adaptive Authentication and Authorization (RAD-AA) Framework. In International Conference on Information Security, Privacy and Digital Forensics (pp. 371-385). Singapore: Springer Nature Singapore.

23. [Lewis 2023] Lewis Golightly, Paolo Modesti, Rémi Garcia, Victor Chang, Securing distributed systems: A survey on access control techniques for cloud, blockchain, IoT and SDN, Cyber Security and Applications, Volume 1, 2023, 100015, ISSN 2772-9184, https://doi.org/10.1016/j.csa.2023.100015.
24. [Lopez 2004] Javier Lopez, Rolf Oppliger, and Günther Pernul. 2004. Authentication and authorization infrastructures (AAIs): a comparative survey. Comput. Secur. 23, 7 (October 2004), 578–590. https://doi.org/10.1016/j.cose.2004.06.013
25. [Bumiller 2023] Anne Bumiller, Stéphanie Challita, Benoit Combemale, Olivier Barais, Nicolas Aillery, et al... On Understanding Context Modelling for Adaptive Authentication Systems. ACM Transactions on Autonomous and Adaptive Systems, 2023, 18 (1), pp.1-35. 10.1145/3582696
26. [Ma 2024] Ma, B., Qian, H. A scalable identity management scheme via blockchain: Identity protection and traceability. Peer-to-Peer Netw. Appl. 18, 1–12 (2025). https://doi.org/10.1007/s12083-024-01866-w
27. [Trnka 2022] Trnka M, Abdelfattah AS, Shrestha A, Coffey M, Cerny T. Systematic Review of Authentication and Authorization Advancements for the Internet of Things. Sensors (Basel). 2022 Feb 10;22(4):1361. doi: 10.3390/s22041361. PMID: 35214259; PMCID: PMC8963074.
28. [Otta 2023] Otta SP, Panda S, Gupta M, Hota C. a Systematic Survey of Multi-Factor Authentication for Cloud Infrastructure. Future Internet. 2023; 15(4):146. https://doi.org/10.3390/fi15040146
29. [Fayad 2018] Achraf Fayad, Badis Hammi, Rida Khatoun. An adaptive authentication and authorization scheme for IoT's gateways: a blockchain based approach. 2018 Third International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC), Oct 2018, Shanghai, China. pp.1-7, 10.1109/SSIC.2018.8556668
30. [Patricia Arias-Cabarcos, Christian Krupitzer, and Christian Becker. 2019]. A Survey on Adaptive Authentication. ACM Comput. Surv. 52, 4, Article 80 (July 2020), 30 pages. https://doi.org/10.1145/3336117
31. [Gutt 2024] Gutt, Federico, Martin Huschka, and Alexander Stolz. "Cascading Effects Analysis Enabled by Semantic Interoperability in the Resilience Data Space.", Position Paper, 2024
32. [Fernandez-Gago 2024] Fernandez-Gago, Carmen, et al. "Trust interoperability in the Internet of Things." Internet of Things 26 (2024): 101226.
33. [More 2023] More, Stefan. "Trust Scheme Interoperability: Connecting Heterogeneous Trust Schemes." Proceedings of the 18th International Conference on Availability, Reliability and Security. 2023.
34. [Ghirmai 2022] Ghirmai, Siem et al. "Self-sovereign identity for trust and interoperability in the metaverse." 2022 IEEE Smartworld, Ubiquitous Intelligence & Computing, Scalable Computing & Communications, Digital Twin, Privacy Computing, Metaverse, Autonomous & Trusted Vehicles (SmartWorld/UIC/ScalCom/DigitalTwin/PriComp/Meta). IEEE, 2022.
35. [Cuñat 2024] Cuñat, Salvador et al. "Secure, Trusted, Privacy-Protected Data Exchange in an-Edge-Cloud Continuum Environment." IoT Edge Intelligence. Cham: Springer Nature Switzerland, 2024. 201-231.
36. [Badirova 2024] Badirova, Aytaj et al. "Towards Robust Trust Frameworks for Data Exchange: A Multidisciplinary Inquiry." Open Identity Summit 2024. Gesellschaft für Informatik eV, 2024.
37. [Abie 2022] Abie, Habtamu, Trenton Schulz, and Reijo Savola. "Adaptive Security and Trust Management for Autonomous Messaging Systems." arXiv preprint arXiv:2203.03559 (2022).
38. [OpenID-Connect] OpenID Connect Core 1.0 Specification https://openid.net/specs/openid-connect-core-1_0.html