Ref. Ares(2025)6212677 - 30/07/2025

Federated decentralized trusted dAta Marketplace for Embedded finance

D4.6 - Operational Models, Business Models and Governance II

Title	D4.6 - Operational Models, Business Models and Governance II
Revision Number	3.0
Task reference	T4.5
Lead Beneficiary	JRC
Responsible	Konstantina Tripodi
Partners	ENG, EUBA, JOT, TRB, UIA
Deliverable Type	DEM
Dissemination Level	PU
Due Date	2025-06-30 [Month 30]
Delivered Date	2025-07-30
Internal Reviewers	IDSA NOVO
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

©FAME Consortium

Revision History

Version	Date	Partners	Description
0.1	2025-05-16	JRC	First version with TOC
0.2	2025-05-30	JRC, EUBA, JOT	First Partners’ Contributions
0.3	2025-06-27	JRC, ENG	Partners’ Contributions
0.4	2025-07-07	JRC, UPRC	Partners’ Contributions
0.5	2025-07-14	JRC, IDSA	Partners’ Contributions
2.0	2025-07-30	JRC, UPRC	QA
3.0	2025-07-30	JRC, UPRC	Version for submission

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

Definitions

Acronyms	Definition
AAI	authentication authorization infrastructure
AI	Artificial Intelligence
AML	Anti Money Laundering
AP	Asset Pricing
API	Application Programming Interface
AWS	Amazon Web Services
BERT	Bidirectional Encoder Representations from Transformers
CBDC	Central Bank Digital Currency
CRUD	Create Retrieve Update Delete - Basic Operations in DBMS
DB	Data Base
DCAT	Data Catalog Vocabulary
DGA	Data Governance Act
EU	European Union
FAIR	Findable Accessible Interoperable Reusable
FAME	Federated decentralized trusted dAta Marketplace for Embedded finance
FDAC	Federated Data Assets Catalogue
FFGA	FAME Federation Governance Application
FGB	FAME Governance Board
GDPR	General Data Protection Regulation
GOV	Operational Governance
HTTP	HyperText Transfer Protocol
ID	Identity
IDS	International Data Spaces
IDSA	International Data Spaces Association
IP	Internet Protocol
ISO	International Organization for Standardization
IT	Information Technology
JWT	JSON Web Token
KYC	Know Your Customer
LC	Learning Center
ML	Machine Learning
MVP	Minimum Viable Product|Platform
OID4VC	OpenID for Verifiable Credentials
PAT	Pricing Advisory Tool

PSP	Payment Service Provider
RA	Reference Architecture
REST	Representational State Transfer
SA	Supervisory Authority
SDK	Software Development Kit
SLA	Service Level Agreement
SQL	Structured Query Language
SSI	Server Side Includes
TID	Trade Account Identifier
UID	User Identifier
UUID	Universally Unique Identifier

Executive Summary

This deliverable, D4.6 ‘’Operational Models, Business Models and Governance II’’, continues the work initiated under Task T4.5 “Business and Operational Models for the FAME Marketplace”. It presents refined business strategies and an updated configuration of the Operational Governance (GOV) module, which is responsible for managing user onboarding, supporting monetization models, and facilitating trading operations within the FAME Marketplace.

The GOV module provides a foundational framework for the administration of the FAME federation, ensuring that all governance and operational functions are executed in a secure, scalable, and efficient manner. It supports various business models—such as subscription, pay-as-you-go (PAYG), pay-as-you-use (PAYU), Data-as-a-Service (DaaS), and freemium—selected for their flexibility and adaptability across different use cases. These models aim to support both the financial sustainability of the platform and the diverse needs of its user base.

Throughout this phase, pilot activities have provided critical validation for the business models and GOV functionalities. The insights gained have informed the refinement of key components, improved integration with other modules, and helped align operational processes with real-world requirements. The GOV module remains structured around four key components:

- Federation Management: Enables registration, onboarding, and administration of federation members.
- User Management: Handles identity management and access controls for platform users.
- User Support: Manages user assistance via asynchronous request handling and ticketing systems.
- Trading Support: Oversees trading permissions, token-based operations, and interaction with the marketplace infrastructure.

On the technical side, the module has been enhanced with improved APIs, standardized workflows, and deployment readiness. Key advancements include containerized deployment architectures, integration with blockchain-based components, and security hardening practices to ensure robustness and scalability.

This deliverable also outlines deployment activities, including the demonstration of the Minimum Viable Product (MVP) and the installation procedures for both server-based and blockchain-based components. These practical guidelines support reproducibility and facilitate broader adoption of the GOV module.

On the technical side, the deliverable details modular components, including federation and user management, trading support, and ecosystem analytics, each supported by updated APIs and architectural refinements. Integration workflows—such as onboarding/offboarding procedures and FDE coin procurement—are specified to ensure seamless interaction between actors in the data federation.

Additionally, the integration of Dataspaces as part of the business model building blocks establishes a secure and interoperable framework for federated data sharing. By enabling flexible governance, usage control, and monetization mechanisms within the Dataspace ecosystem, the platform ensures sustainable and trustworthy data exchange aligned with evolving marketplace needs and regulatory frameworks.

A demonstrator module has been deployed, showcasing full system capabilities, containerized server-based software, security features, and Open API documentation. Validation results confirm system

robustness, interoperability, and performance compliance. This deliverable supports the transition toward scalable, federated data marketplaces that balance trust, value exchange, and decentralization.

Moving forward, the work will focus on further extending the business logic, enhancing the pricing mechanism with dynamic data insights, and continuing the alignment of the operational framework with pilot needs. The results presented here contribute to ensuring that the FAME Marketplace remains agile, robust, and responsive to its evolving ecosystem.

Table of Contents

List of Figures

List of Tables

Introduction

The second version of deliverable D4.6-Operational Models, Business Models and Governance II presents refined business models and updated configuration of the Operational Governance (GOV) module. As part of the broader activities under Task T4.5, this document reflects the progress made since the initial iteration and highlights the outcomes of validation efforts, integration with technical components, and feedback from pilot deployments.

The GOV module is designed to provide the services necessary to support user registration, federated identity management, and interaction with various business models, including pay-as-you-go, pay-as-you-use, and Data-as-a-Service (DaaS). The shift of SLA management responsibilities to Task T4.1-Decentralized Data Provenance and Traceability remains in effect.

The document builds on the hybrid business model approach introduced previously, refining it through the application of a business model selection methodology and alignment with operational feedback. The GOV module continues to serve as a foundational element of the marketplace infrastructure, supporting both technical processes and governance workflows. Emphasis has been placed on creating a governance environment that upholds ethical standards, transparency, and accountability while ensuring interoperability and efficiency.

1.1 Scope and Objectives

The objective of this deliverable is to consolidate and validate the business models defined for the FAME Marketplace and to update the operational and governance frameworks accordingly. It presents the enhanced configuration of the GOV module, which includes the administration of federation members, user lifecycle management, user support mechanisms, and the facilitation of trading transactions. These updates aim to ensure that the marketplace is both operationally viable and commercially sustainable.

The deliverable also aims to support broader exploitation and sustainability goals by aligning operational practices with the evolving needs of the federated ecosystem. Strategic emphasis is placed on creating a seamless user experience, fostering interoperability among actors, and leveraging data as a driver for monetization .

1.2 Summary of Previous Insights and Evolving Context

Task T4.5 remains interconnected with other key tasks in WP4 and beyond. It leverages initial design inputs and SLA mechanisms from T4.1 “Decentralized Data Provenance and Traceability”, and adopts monetization strategies and accounting mechanisms developed under T4.2 “Accounting, Trading, Pricing and Monetization Schemes”. Interoperability with smart contracts and programmable logic is supported through T4.3 “Smart Contracts for Programmable Trading and Monetization”, while semantic-based data discovery insights are derived from T4.4 “Semantic Search for Trading and Valuation of Data Assets”.

Technical alignment with T3.1 “Federated AAI Infrastructure” and T3.2 “Unified Security Policy Management” supports secure authentication and authorization processes. Finally, the business model outputs continue to serve as input to T7.3 “Exploitation and Sustainability Planning”, which addresses long-term viability and uptake of the FAME Marketplace.

1.3 Structure of the Deliverable

This deliverable is structured into six main chapters:

- *Chapter 1: Introduction* provides a general overview of the objectives, insights from related tasks, and the structure of the document.
- *Chapter 2: Data Marketplace Business Models* updates the business models explored, revises the evaluation framework, and reflects validation findings from pilot use cases.
- *Chapter 3: Dataspaces and Business Model Building Blocks* outlines how business models are applied within dataspaces to enable trusted, value-driven data exchange in federated environments. It details the key technical and organizational building blocks needed to support monetization, interoperability, and governance across diverse actors.
- *Chapter 4: Module Overview* presents the current structure and functionalities of the GOV module, with emphasis on integration into the FAME architecture and updated workflows.
- *Chapter 5: Components Specification* describes the technical specifications of the module's components, including interfaces and dependencies.
- *Chapter 6: Module Demonstration* documents the deployment and demonstration of the updated GOV MVP, including configuration guides for server-based and blockchain-based environments.
- *Chapter 7: Conclusion* summarizes the main findings of the deliverable, outlines the next steps in implementation and scaling, and highlights areas for further enhancement.

1.4 Summary of Changes

Compared with the previous version of this deliverable (D4.3- "Operational Models, Business Models and Governance"), this updated document introduces significant enhancements in both structure and content to better reflect the progress made in the project. A new focus has been added on how business models are applied within federated data ecosystems, providing a detailed explanation of the key technical and organizational components necessary to support trusted, value-driven data exchange, including monetization, governance, and interoperability among diverse participants.

In parallel, the platform architecture has advanced significantly, with multiple core modules undergoing targeted redesigns and improvements to increase modularity, performance, and integration flexibility. Federation management now includes robust member migration features, more transparent status tracking mechanisms, and a refined separation between public-facing and internal operations—enhancing usability for operators and ecosystem participants alike. User management capabilities have also matured, with new functions such as user reinitiation, enhanced credential integration, and a more comprehensive approach to lifecycle management, all contributing to a smoother administrative experience.

1.4.1 Major Updates Since Previous Release

- **Federation Management**: Enhanced with member migration capabilities, improved status tracking, and better separation of public/internal operations.
- **User Management**: Added user reinvitation functionality, improved credential management integration, and enhanced user lifecycle operations.
- **Trading Support**: Complete architectural overhaul with streamlined account management, simplified API surface, and improved integration patterns.
- **User Support**: Request queue functionality implemented; full ticketing system remains planned for future release.
- **Ecosystem Statistics**: New component providing platform activity metrics.

1.4.2 API Architecture Improvements

- **Consistent Error Handling**: Standardized HTTP status codes and error responses across all endpoints
- **Enhanced Security**: Improved access control patterns with role-based permissions
- **Better Separation of Concerns**: Clear distinction between public and internal APIs
- **Asynchronous Processing**: Consistent patterns for long-running operations
- **Improved Data Validation**: Enhanced input validation and type safety

1.4.3 Implementation Status

- **Fully Implemented**: Federation Management, User Management, Trading Support, Ecosystem Statistics
- **Partially Implemented**: User Support (request queue only)

Planned: Complete User Support ticketing system

The platform continues to evolve following agile development principles, with ongoing enhancements based on pilot feedback and operational requirements.

Overall, this deliverable reflects the outcomes of agile, iterative development driven by real-world feedback and emerging operational needs. With most core components fully implemented and others well underway, the platform is now significantly more robust, coherent, and prepared for broader deployment and scaling across federated data environments.

2 Refined Data Marketplace Business Models

This chapter focuses on updating the business strategy of the marketplace by revisiting models, pricing schemes, and external market conditions.

2.1 Updated Competitor & Market Landscape

The data economy continues to evolve rapidly, influenced by technological advances, regulatory developments, and shifting user needs. The FAME Marketplace operates within a highly dynamic environment, where competitors offer various degrees of centralization, specialization, and financial models.

Key market trends:

- **Emergence of specialized data marketplaces** targeting vertical sectors such as finance, healthcare, and manufacturing.
- **Decentralization and tokenization** gaining momentum as trust, sovereignty, and compliance become critical for cross-border data transactions.
- **Integration of AI-powered services** (such as FAME's PAT - Pricing Advisory Tool) into data valuation and trading strategies.
- **Regulatory tightening** in Europe (e.g. GDPR, DGA, Data Act, FiDA, PSD3, DORA) reshaping the boundaries of lawful data sharing and monetization.

Based on market knowledge and alignment with FAME Scope – federated marketplaces for Embedded Finance (EmFi), the main competitors are identified as follows:

- Amazon AWS Data Exchange → a global leader for centralized data exchange models.
- Snowflake Marketplace → very active in enterprise SaaS-based data sharing.
- Ocean Protocol → globally recognized decentralized data marketplace protocol, aligns with FAME's decentralized ambition.
- Datum → more niche but often cited in decentralized personal data ownership spaces.

The table below provides an overview of how key competitors are positioned within the market, considering factors such as value proposition, technological capabilities, customer segments, and strategic focus. This comparison helps identify market gaps, differentiation opportunities, and emerging trends shaping competitive dynamics.

Competitor	Model	Specialization	Key Differentiator
Amazon AWS Data Exchange	Centralized	Broad enterprise	Coupled compute/storage
Snowflake Marketplace	Centralized	Enterprise SaaS	Seamless data warehousing integration
Ocean Protocol	Decentralized	Cross-domain	Blockchain-based, open protocol
Datum	Decentralized	Personal data	Individual control and monetization
FAME	Federated, hybrid	Financial, enterprise data	Regulatory compliance, embedded finance,

			tokenization, sovereign control

Competitor	Model	Specialization	Key Differentiator
Amazon AWS Data Exchange	Centralized	Broad enterprise	Coupled compute/storage
Snowflake Marketplace	Centralized	Enterprise SaaS	Seamless data warehousing integration
Ocean Protocol	Decentralized	Cross-domain	Blockchain-based, open protocol
Datum	Decentralized	Personal data	Individual control and monetization

Table 1 Competitor Positioning Overview Across Key Strategic Dimensions

FAME differentiates itself by combining a federated architecture with embedded finance and strong regulatory alignment, particularly suitable for financial data markets in Europe. In details, these are the most important differentiation elements from competitors on the market:

- **FAME as a Federated Data Marketplace**: FAME directly addresses the recognized limitations of centralized cloud marketplaces by adopting a federated, decentralized, and interoperable design fully aligned with GAIA-X, IDSA and EU Digital Market goals.
- **Cross-sector pilot-driven approach**: FAME integrates data marketplaces across domains (financial services, supply chains, IoT, e-commerce, healthcare, etc.), building stronger market resilience by combining multiple verticals.
- **Ecosystem-centric competition**: Beyond pure data trading, FAME includes platforms offering cross-border regulatory compliance, trusted AI analytics, and embedded finance capabilities.

Key New Differentiators of FAME:

- Secure and compliant exchange for *Embedded Finance* (EmFi) use cases.
- Programmable trading mechanisms using blockchain tokenization, NFTs, and smart-contract-based escrow.
- Federated Learning and AI integration with energy-efficient and privacy-preserving analytics.
- A dedicated Learning Centre (LC) to support market education and adoption

2.2 Pricing, Trading Strategies & Monetization Refinements

Pricing digital assets in the FAME marketplace requires a careful balance between the economic value of information and the diverse ways users engage with and consume data. Digital assets vary significantly in structure, usage patterns, and strategic importance, which necessitates a flexible pricing framework that can accommodate various business models. FAME addresses this challenge by implementing a set of base pricing strategies tailored to different transaction modes. These strategies ensure fairness, scalability, and compatibility with both subscription and pay-as-you-go (PAYG) access models. Digital assets on the marketplace are modelled as network by deriving similarity between them. The base price, once computed, is further refined using a weighted average that incorporates both the number of transactions and the network-based eigenvector centrality (EVC), thus reflecting the relative importance of each transaction within the data ecosystem.

- **Subscription Model** - in the subscription model, users pay a fixed price for time-bound access to a digital asset. To calculate the base price for this model, we divide the total price of the offering by the number of access days, then normalize it by the number of subscribers and an adjustment factor accounting for time value. This method ensures that the cost is fairly distributed across time and users, making it ideal for regularly updated data streams. It is particularly well suited to continuous data or platforms with regular updates, as the formulation adjusts for value decay or growth.
- **PAYG per download -** this approach is intended for scenarios where users are charged per download. The base price is determined by dividing the total pre-paid amount by the expected number of downloads. To reflect realistic usage patterns, this result is then adjusted by a yearly working day factor. It is best suited to assets accessed in discrete download events, such as research datasets or media files.
- **PAYG per volume -** For digital assets accessed based on usage volume, such as API calls or bandwidth, the base price is calculated by dividing the total pre-paid amount by the proportion of the total volume that corresponds to the size of the asset. This value is then normalized using a standard yearly activity factor to account for realistic distribution over time. This strategy is suitable for services where data consumption is measured in units like megabytes or requests.
- **Buy-the-Asset-** this straightforward model applies when the user purchases the full digital asset in one transaction. The base price in this case is equal to the total price of the offering, reflecting the full value of the digital asset. This model is appropriate for static, high-value digital assets with long-term relevance and no expected updates.
- **Learning Centre monetization**: Training, certification, and advisory services as revenue streams for onboarding data providers and consumers.

These five strategies provide the FAME marketplace with a robust and adaptive pricing backbone, ensuring equitable value distribution across different usage modes. The final base price used in transactions is further adjusted by applying a weighted average that incorporates transactional volume and network-theoretic conceptualization of eigenvector centrality, ensuring prices remain aligned with market dynamics and user influence.

This flexible and adaptive pricing structure allows FAME to simultaneously accommodate various participant sizes, transaction volumes, and compliance requirements.
From the pilot inputs gained so far, we can now evaluate differentiated pricing schemes for:

- Financial KYC & AML datasets.
- IoT industrial sensor data.

- Healthcare compliance datasets.
- Retail and behavioural analytics.

2.3 Enhanced Business Models List

In response to evolving ecosystem feedback and rapid technological advancements, organizations are continuously adapting their business models. Some models remain actively employed to capitalize on current trends and customer needs, while others are being gradually phased out due to shifts in market demands or technological obsolescence. The table below summarizes these business models, indicating which are currently active and which are in the process of being phased.

Business Model	Status	Remarks
Federated Marketplace Licensing	Federated Marketplace Licensing	Licensing framework for operators of fed instances.
Transaction-based Marketplace	Retained	Forms the core of the trading engine.
Subscription-based Access	Enhanced	Provides predictable revenue for long-term engagement.
Tokenization & Smart Contracts	Expanded	Enables micro-payments, escrow, programmable value flows.
Revenue Sharing	Retained	Shared benefits with data providers and service contributors.
Embedded Financial Services	Introduced	Supports fiat and tokenized payments, increasing accessibility.
Embedded Finance Integration	Embedded Finance Integration	Deep integration with EmFi infrastructure financial automation.
Decentralized Identity & SSI	Active	Strengthens compliance, privacy, and trust.
Marketplace Connector Ecosystem	Expanded	Enables third-party integrations, scaling interoperability.
Data Custodian Services	Under Development	Adds secure asset storage and regulatory compliance capabilities.

Table 2 Overview of Active and Phasing-Out Business Models Based on Ecosystem Feedback and Technological Developments

The following Revised Business Models Evaluation Matrix presents a comprehensive and updated framework for assessing the relevance and feasibility of each business model. This evaluation incorporates key criteria derived from recent ecosystem feedback, technological advancements, and market trends.

Business Model	Market Demand	Technical Feasibility	Regulatory Compliance	Revenue Potential	Scalability	Business Model

Transaction-based Marketplace	High	High	High	Medium	High	Transaction-based Marketplace
Subscription-based Access	High	High	High	High	High	Subscription-based Access
Tokenization & Smart Contracts	Growing	Medium	High (with proper design)	High	High	Tokenization & Smart Contracts
Revenue Sharing	High	High	High	High	High	Revenue Sharing
Embedded Finance	Growing	Medium-High	Complex, but manageable	High	Medium-High	Embedded Finance
Decentralized Identity (EUDI, SSI)	Increasing	Medium	High	Indirect	High	Decentralized Identity (EUDI, SSI)
Marketplace Connector Ecosystem	Growing	Medium	Medium-High	Indirect	High	Marketplace Connector Ecosystem
Data Custodian Services	Emerging	Medium	High	Moderate	Medium	Data Custodian Services
Business Model	Market Demand	Technical Feasibility	Regulatory Compliance	Revenue Potential	Scalability	Business Model
Transaction-based Marketplace	High	High	High	Medium	High	Transaction-based Marketplace

Table 3 Revised Business Models Evaluation Matrix

2.4 Pilots' Feedback and Extended Validation

The original proposal highlights seven pilots across domains. These pilots directly feed into business model evolution:

- Financial regulatory compliance pilots (AML/KYC)
- Retail customer behaviour analytics (Retail & eCommerce)
- IoT and industrial asset data trading
- Healthcare data sharing pilots with privacy guarantees
- Decentralized AI/ML model marketplaces
- Supply chain traceability pilots
- Smart city data sharing scenarios

The FAME Marketplace underwent extensive validation through its seven pilot implementations across multiple sectors, validating both the technical infrastructure and the business framework across heterogeneous ecosystems.

Key findings from pilots:

- **Financial Sector (AML/KYC, Embedded Finance):**

    - Strong demand for compliant cross-border financial data exchange.
    - The tokenized payment model (using smart contracts and programmable payment flows) demonstrated operational efficiency, trust, and real-time auditability.
    - The federated identity and consent management models were validated, improving data privacy compliance.
- **Retail & e-Commerce:**
    - Retailers and e-commerce platforms benefited from real-time access to consumer behaviour data, enabling more accurate personalization.
    - Data bundling services (DaaS) gained particular interest, where both raw datasets and pre-processed analytics were offered under tiered pricing.
- **IoT & Industry 4.0:**
    - IoT sensor data marketplaces for predictive maintenance, industrial asset monitoring, and supply chain optimization proved viable.
    - Pilots validated marketplace connectors and semantic interoperability layers for industrial data, supporting federated deployment models.
- **Healthcare & Privacy-Sensitive Domains:**
    - Federated data exchange models were effective in enabling cross-institutional data sharing while preserving privacy and confidentiality.
    - Compliance-by-design mechanisms (GDPR, Data Act) ensured secure, lawful exchange.
- **Supply Chain & Logistics:**
    - Distributed ledger technology supported traceability, data escrow, and conditional release models.
    - Multiple stakeholders benefited from transparent, tamper-proof transactional records.

Extended Validation Insights:

- The pilots confirmed the robustness of the **hybrid architecture**, blending centralized orchestration with decentralized autonomy.
- The **Pricing Advisory Tool (PAT)** was validated as a powerful component for dynamic pricing and negotiation support.
- Stakeholders favoured flexible engagement models (subscription-based + pay-per-use).
- The importance of a **Learning Centre (LC)** was reinforced to facilitate onboarding, certification, and continuous business model adoption.

The extended validation phase demonstrated that FAME's federated architecture is adaptable across multiple verticals, with high scalability potential and cross-sector applicability.

2.5 Finalized Hybrid Business Model & Implementation Variants

Following iterative refinements through pilots and stakeholder feedback, the FAME Business Model has matured into a flexible **Hybrid Federated Marketplace Framework** combining multiple monetization, operational, and regulatory layers.

2.5.1 Core Components

The table below summarizes the core components that enable platform functionality, monetization, and compliance. These elements support flexible business models, secure transactions, and scalable ecosystem engagement.

Component	Description
Federated Licensing	API-based integration for third-party marketplaces

Transaction Trading	Core pay-per-use exchange engine
Subscription Models	Tiered access for multiple customer classes
Prising Advisory Tool (PAT)	Data-driven tool for pricing optimization
Tokenized Payments	NFTs and smart contracts managing programmable asset transfers
Revenue Sharing	Collaborative revenue split with providers
Custodial Trust Services	Trusted escrow, multi-party authorization, regulatory compliance assurance
Embedded Finance	Integrated fiat and token-based settlements
Certification & Learning	Monetization of training, onboarding, compliance services

Table 4 Overview of Core Components Enabling Platform Services and Monetization

2.5.2 Deployment Variants

The table below outlines the key deployment variants; each tailored to specific target segments and application scenarios. These models address varying compliance requirements, scalability needs, and industry-specific use cases.

Variant	Target Segment	Application Scenario
Enterprise SaaS Model	Finance, critical infrastructure	Private deployments for high-compliance industries
Vertical Federated Hubs	Healthcare, IoT, Manufacturing	Federated nodes interconnecting sector-specific partners
Open Federated Marketplace	SMEs, start-ups, academia	Public connectors lowering entry barriers
B2B Data Exchange Connectors	Supply chains, IoT sensors	Lightweight industrial data exchange nodes

Table 5 Deployment Variants by Target Segment and Application Scenario

The hybrid model ensures scalability, regulatory alignment, cross-sector flexibility, and diversified revenue generation — positioning FAME uniquely in the European data economy.

2.6 Pricing Advisory Tool

Price suggestions for digital assets in FAME are generated via Pricing Advisory Tool (PAT). PAT determines the suggested price of a digital asset by combining two components: a price based on questionnaire responses and the selected business model (denoted as **P1**), and a dynamic component derived from market demand and the pricing of similar digital assets (denoted as **P2**). The final recommended price is calculated as the geometric average of these two components.

- **Component P1: Questionnaire and Business Model-Based Pricing**
  The base price (P1) of a focal digital asset is initially established from questionnaire responses, which encapsulate characteristics and value propositions of the digital asset as well as specific business model information—such as whether the offering follows a subscription model, a pay-as-you-go (PAYG) scheme based on downloads or volume, or a one-time purchase.
  - **Subscription model:** Price is calculated based on the number of subscription days.
  - **PAYG – Downloads**: Pricing considers the number of pre-paid downloads.
  - **PAYG – Volume:** The pre-paid volume and the digital asset's actual size are essential.
  - **Buy the Digital Asset:** A fixed price is assigned to the outright purchase.
- **Component P2: Dynamic Market-Based Pricing**
  To determine the market-responsive price (P2), a multistep procedure is followed to analyze the pricing of comparable digital assets.
  - **Step 1 – get a subset of all digital asset traded in the market**
    A subset of similar digital assets (S) is selected based on semantic similarity determined by language model BERT (Bidirectional Encoder Representations From Transformers). BERT embeddings are used to ensure a similarity threshold, filtering digital assets that are conceptually and contextually aligned with the focal digital asset.
  - **Step 2 – create network of the digital assets in subset**
    A network of digital assets in subset is created by defining edges based on a second similarity measure—questionnaire-based similarity (QBS). An edge is formed if two digital assets have QBS similarity greater than predefined threshold.
    To capture market dynamics, a second price component is generated based on how similar digital asset are priced and how they perform in terms of sales. The process starts by identifying digital asset in the market that are similar in domain or content. A language model is used to measure semantic similarity, and only those digital assets that meet a certain similarity threshold are included in the analysis. Next, a network is built from this subset of similar digital asset, where connections between them are established based on how closely they align in terms of questionnaire-based attributes. This network is then analysed to determine the influence or relevance of each digital asset within it. For those digital assets closely related to the focal digital asset, additional digital is collected—including their prices, number of sales, and their centrality within the network. Each digital asset's influence is weighted accordingly to calculate an average benchmark price. From this, the dynamic price is adapted to match the specific business model of the focal digital asset, whether it's a subscription, PAYG (downloads or volume), or a direct sale.

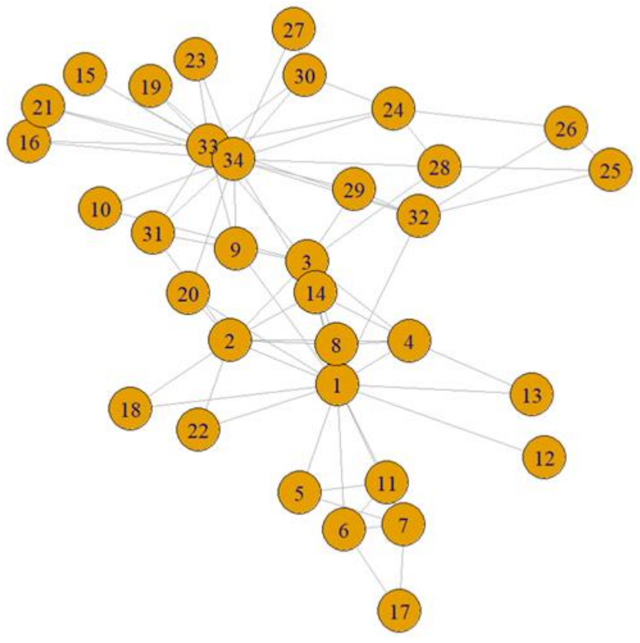


Figure 1 Step 2 – Network Construction and Dynamic Pricing Based on Questionnaire-Based Similarity and Market Performance

- **Step 3 – compute eigenvector centrality**
  Eigenvector centrality (EVC) is calculated for each digital asset in the network, indicating its importance or influence in the ecosystem of similar digital assets.
- **Step 4 – the price P2 will be computed**
  The dynamic price uses information from the set of connected digital assets in the network—those with QBS similarity over threshold. Each offering within connected digital assets in the network contributes to the pricing based on three factors:
  - its base price (depending on its business model),
  - the number of transactions (sales), and
  - its EVC score.

  From this, a dynamic price for the focal digital asset is computed for each business model similarly to the approach in P1.

Final Price Calculation and Visualization

The PAT tool synthesizes both pricing components by computing their geometric average, providing a balanced recommended price that considers intrinsic data value and market trends.

In addition to computing prices for all connected digital assets, the following data are required:

- Their eigenvector centrality (computed in step 3)
- Their number of transactions (sales), and
- Their business model-based base price

For visualization purposes, normalized prices are presented for different offerings. Then, based on the number of transactions and EVC, a weighted average of the base price is computed. Afterward:

1. The suggested price is derived from this weighted average.
2. A second price is calculated for the selected business models.

Finally, the recommended price of the focal digital asset is computed as a geometric average of the above both prices.

3 Data Spaces and Business Model Building Blocks

This chapter offers a detailed explanation of how technical infrastructure and business model innovation interact in federated dataspaces, as demonstrated and verified in the GOV module. It describes the precise mechanisms that facilitate long-term, trust-based data sharing, building on the findings from the pilot and platform improvements. These mechanisms include usage-based monetization (e.g., token mechanisms), identity and access control, and standardized contract enforcement.By specifying interoperable components including data usage control layers, policy enforcement modules, and accounting functions, the chapter directly supports the design of the hybrid business model. It also explains how transparent governance (such as onboarding lifecycles and policy audits) and trust anchors (such as verified metadata and credentialed identities) allow participants—both data suppliers and consumers—to engage in safe and predictable interactions. By doing this, it creates a useful link between strategy and deployment by connecting the technological architecture.

3.1 Purpose of the Building Block in FAME Federated Marketplace

The purpose of this building block is to operationalize sustainable business models within FAME federated data ecosystems by combining governance logic with enforceable technical rules. It ensures that data exchanges occur under transparent conditions, governed by policy-enforced contracts, and monetized via flexible accounting models (e.g., pay-per-use, subscriptions, freemium). The building block acts as a bridge between data providers, consumers, and intermediaries, enabling revenue-generating interactions while preserving compliance, trust, and discoverability.

The implemented building block is an essential foundational component to build the EU Common Data Space Strategy. *DSSC reference:* Building Block Overview– supports a modular framework that structures data spaces into manageable components across business, governance, legal, technical, and trust categories to enable scalable and interoperable implementations.

FAME reference: FAME D2.6-Technical Specifications and Platform Architecture II (Section 8.2.3 FAME SA Mapping to DSSC Building Blocks , 2024)

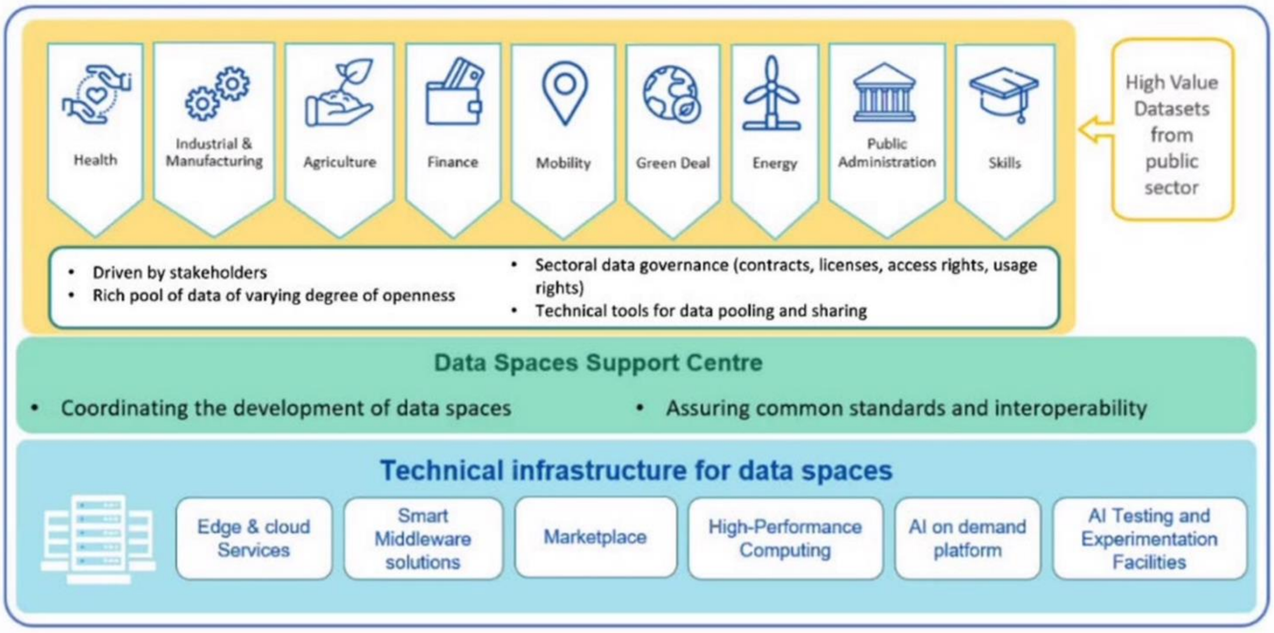

Health
Industrial & Manufacturing
Agriculture
Finance
Mobility
Green Deal
Energy
Public Administration
Skills
High Value Datasets from public sector
Driven by stakeholders
Rich pool of data of varying degree of openness
Sectoral data governance (contracts, licenses, access rights, usage rights)
Technical tools for data pooling and sharing
Data Spaces Support Centre
Coordinating the development of data spaces
Assuring common standards and interoperability
Technical infrastructure for data spaces
Edge & cloud Services
Smart Middleware solutions
Marketplace
High-Performance Computing
AI on demand platform
AI Testing and Experimentation Facilities

Figure 2 Overview of the European Data Spaces Ecosystem: Sectoral Focus, Governance Mechanisms, and Supporting Infrastructure

As shown in the overall Data Space Strategy diagram, a Marketplace is considered as an essential Technical Infrastructure component, allowing– together with a smart middleware solution and other services – the easy exchange of data between independent data space participants in multiple sectors.

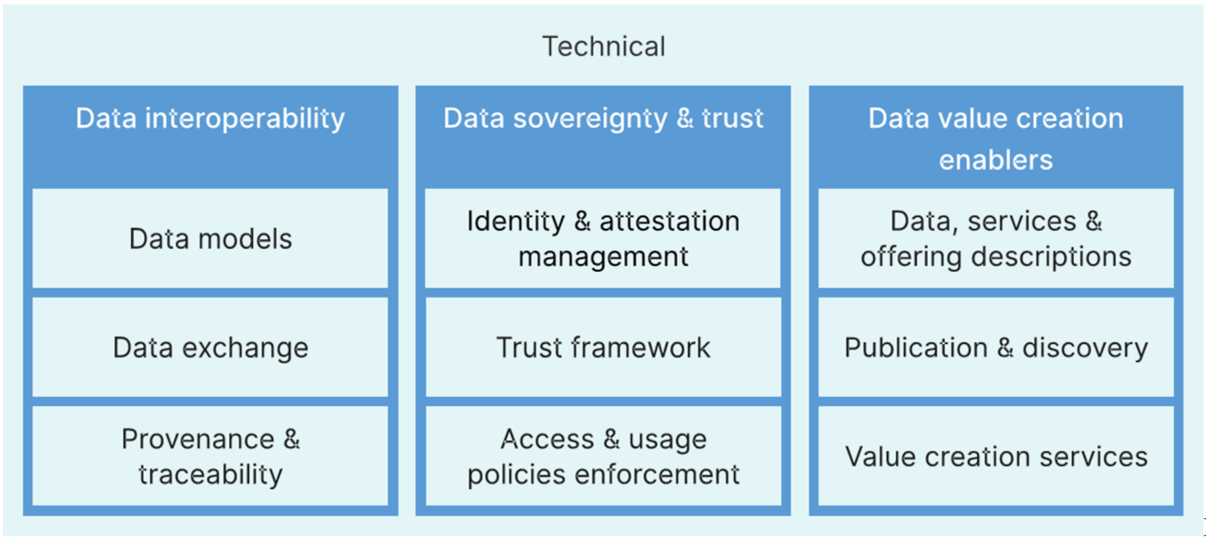

Technical
Data interoperability
Data models
Data exchange
Provenance & traceability
Data sovereignty & trust
Identity & attestation management
Trust framework
Access & usage policies enforcement
Data value creation enablers
Data, services & offering descriptions
Publication & discovery
Value creation services

Figure 3 Core Technical Building Blocks for Data Space Enablement

3.2 Business Model Concepts Applied to Data Spaces

This section maps key business concepts to their realization within dataspaces, focusing on three foundational areas:

3.2.1 Data Intermediaries and Actor Roles

The dataspace framework supports a triad of roles: data providers, data consumers, and intermediaries. Intermediaries (e.g., brokers or marketplaces) facilitate trust, access control, and billing operations. Each role is technically modeled through identity credentials, access policies, and role-based workflows in the GOV module.

DSSC reference: Identity & Attestation Management – supports role-based workflows, credential issuance, and identity verification for all actors in the dataspace.

The provided component delivers tooling for intermediaries in a Data Space to bring together data providers and data consumers and enables payments to happen, i.e. the exchange of value, represented by a payment token, against a data asset token, representing the bought access to the data asset. A prerequisite for this marketplace to function well is a robust Identity Management system. In a Data Space context, which is an ecosystem of loosely interconnected actors, each of them remaining in full control over their own data assets, the identities are set up according the principles of Self-Sovereign Identity (SSI), and with Verifiable Credentials (VCs) being used to certify the actors on their role in the ecosystem and to provide them keys that uniquely identify them, not only to hand out certifications but also to be active in the marketplace as buyer or seller of the data assets they are consuming or providing. *DSSC reference:* Identity & Attestation Management – for setting up decentralized identities and issuing verifiable credentials.

3.2.2 Usage-based Monetization and Token Integration

Token-based logic underpins usage monetization, allowing for FDE coin consumption based on dynamic metrics such as data volume, query frequency, or service access. Smart accounting events are triggered via usage logs, with payment flows enforced through programmable rules embedded in the GOV's trading and accounting subsystems.

The public-private key pairs that are created and used for certifying the identity of all actors are used to create a wallet for these actors, which can hold both payment tokens and data access tokens. Payment tokens are assets that represent financial value. It is up to the federation to manage these payment tokens, it might make sense in the future to replace this token by a stablecoin or a CBDC (Central Bank Digital Currency, like digital EUR).

Data Access Tokens (DATs) represent the access to a digital asset described in the metadata linked to the DAT.

3.2.3 Data Productization and Trust Anchors

Data assets are transformed into exchangeable products, described by rich metadata, usage conditions, and licensing terms. Each data product is anchored in trust frameworks (e.g., verified credentials, legal contracts) and integrated into registries that allow semantic search and lifecycle tracking.

DSSC Alignment → Business Enablement Services: The DSSC Blueprint v2.0 includes Section 2.1.6 – Business Enabling Services and 2.2.1 – Example: Data Marketplace, which align with how FAME supports publication, discovery, and marketplace operations

A preliminary step before being able to trade data asset in the marketplace, the asset needs to be published in the FDAC (FAME catalogue), and an offer must be created, specifying the modalities for the trade.

DSSC reference: Data Value Creation Enablers

- *Data, Services & Offering Descriptions – supports semantic descriptions of datasets and services.*
- *Publication & Discovery – aligns with the FDAC to ensure assets are findable and accessible under FAIR principles.*

Value Creation Services – connects with monetization logic and dynamic asset offerings.

3.3 Key Elements and Their Technical Functions

This section details the modular components that support the enforcement and execution of federated business models:

3.3.1 Data Contract and Usage Control Layer

Supports formalized agreements through smart contracts or usage policies. Contracts define authorized actions, consumption limits, expiration rules, and downstream restrictions. Runtime enforcement is implemented via policy engines (e.g., PDPs/PAPs) integrated with access logs and audit trails.

3.3.2 Identity, Access, and Policy Enforcement

Federated identity management is implemented using decentralized identifiers (DIDs) and verifiable credentials (VCs). Access to services and datasets is managed via policy enforcement points (PEPs) that interpret contextual and credential-based access control rules, aligning with zero-trust architecture principles.

3.3.3 Monetization and Accounting Mechanisms

Tracks service and data consumption events, correlates usage with pricing models, and triggers automated accounting processes. Supports event-based token deduction, invoice generation, and revenue splitting logic. Integrated with wallet systems and FDE coin infrastructure.

3.3.4 Metadata Management and Discoverability

Ensures that all data products are described using standard semantic models (e.g., DCAT-AP, IDS Information Models). Products are indexed in a federated catalog with support for search, ranking, and classification. Metadata includes policy tags, provenance, and service descriptors.

3.3.5 Interoperability and Cross-System Mappings

Enables integration with external systems and heterogeneous data sources. Supports semantic mediation, protocol translation (e.g., IDS connectors, GAIA-X compliance), and standard API exposure.

3.4 Dependencies and Integration with Other Modules

The building block directly interacts with:

- Federation Management: For onboarding actors and distributing roles and credentials.
- Trading Support: To validate token flows, execute smart contracts, and log transactions.
- Blockchain-based Deployment: To anchor agreements, ensure tamper-proof execution, and enable auditability.

- User Management: To manage fine-grained access control based on actor attributes and verified identities.

Metadata and Catalog Components: To ensure semantic alignment of data products with pricing and usage metadata.

3.5 Glossary of Dataspace Terms

A curated glossary is included to clarify terms such as:

- Dataspace: A federated system enabling sovereign, policy-driven data sharing among multiple actors.
- Data Product: A discoverable dataset offered with metadata, policies, and usage contracts.
- Tokenization: Process of representing value (e.g., access rights, payment) via cryptographic tokens.
- PAYU: Pay-As-You-Use, a monetization model tied to the actual consumption of data.
- Trust Anchor: A verifiable credential or cryptographic proof ensuring the authenticity of actors or datasets.
- Smart Contract: Self-executing agreements encoded as scripts on a distributed ledger.
- Verifiable Credential: A digital statement that is cryptographically signed and verifiable.
- Usage Policy: A set of conditions defining how a data asset may be accessed and used.
- Data Product: A standardized, discoverable data asset or service offering available for exchange.
- Intermediary: A trusted entity facilitating governance, access, or monetization between actors.

ABAC: Attribute-Based Access Control.PDP/PEP: Policy Decision/Enforcement Points for runtime access control enforcement.

4 Module Overview

The Operational Governance (GOV) module of the FAME Platform is, as its name implies, the entry point for all management and governance functions needed for operating an instance of the FAME federation. There are six types of *information entity* that are managed by the GOV module:

- **Federation Member** – Organizations that are members of the FAME federation.
- **Onboarding Authority (OA)** – A new entity type that registers Decentralized Identifiers (DIDs) used for issuing Verifiable Onboarding Credentials (VOCs) to users. Each Federation Members may register their own DID to act as a trusted OA and autonomously onboard affiliated users, or delegate this role to another Federation Member . This defines a many-to-one relationship with Federation Members.
- **User** – Individuals using the FAME platform. Typically, users are onboarded and managed externally by Federation Members acting as Onboarding Authority. However, the platform also supports internally managed users who may be granted special privileges like ADM ("administrator") and SUP ("user support") by a special kind of entity, classified as Root Authority.
- **Support Ticket** – Requests for assistance submitted by users, processed asynchronously by a human operator.
- **User Request** – Requests submitted by client software, processed asynchronously by a service endpoint.
- **Trading Account** – Blockchain accounts used for trading on the FAME marketplace.

Each of these entity types is supported by a dedicated *database*, where the relevant information is maintained by the GOV module, which provides several *operations* -- i.e., service calls defined by the GOV API -- that users can execute to store, retrieve, or modify information. The Onboarding Authority model provides flexible delegation options: when a Federation Member joins the FAME federation, they can choose to register their own DID and become a trusted OA capable of autonomously onboarding Users affiliated with their organization. Alternatively, they can delegate user onboarding responsibilities to another Federation Member that already operates as a trusted OA. This delegation model ensures that all Federation Members can participate in user onboarding while maintaining the security and trust requirements of the FAME ecosystem.

Blacklisting vs. Offboarding

The GOV module implements a robust entity lifecycle management system that distinguishes between temporary suspension (blacklisting) and permanent removal (offboarding):

- Blacklisting is a reversible mechanism that temporarily suspends the validity of entities (Federation Members, Users, and Onboarding Authorities) without cascading effects or permanent data changes. This allows administrators to quickly respond to security concerns, policy violations, or temporary operational issues while preserving the ability to restore access when appropriate.
- Offboarding, in contrast, is irreversible and triggers cascading deactivation of all dependent entities - for example, offboarding a Federation Member automatically deactivates all associated OAs, Users, and Trading Accounts.

This dual mechanism provides operational flexibility (blacklisting) and definitive enforcement (offboarding) depending on the scenario.

Some operations also have *side effects*, meaning that they can reach beyond GOV-managed information and impact on other modules; for instance, enrolling a Trading Account (see the operation list below) implies *granting transaction permission* to that account on the FAME Blockchain Infrastructure (which is now done by the Provenance & Tracing module -- see deliverable D4.1).

All operations follow the REST approach, and thus can be executed in any order at any time. The authorization rules are operation-specific, but always match one of three options: admin users only, any authenticated users, anyone.

At the time of writing, the GOV module is designed to implement the operations listed in Section 4. As FAME adopts an agile approach to design and development, *these may change in the next release of the software*.

Significant Updates: This updated version incorporates several major enhancements including integration with external governance applications (notably the FAME Federation Governance Application - FFGA), comprehensive offboarding capabilities for both members and users, improved blockchain integration patterns through the P&T module, flexible blacklisting mechanisms for temporary entity suspension, ecosystem statistics reporting for platform monitoring and analytics, and API support for automated payment processing through Digital Euro integration (implementation pending).

From a software architecture point of view, all the operations are implemented by four distinct components: **Federation Management** (deals with Federation Members), **User Management** (deals with Users and includes the OID4VCI implementation for issuing VOCs), **User Support** (deals with Support Tickets and User Requests), and **Trading Support** (deals with Trading Accounts). Individual operations are thoroughly documented in Section 4.

At the time of writing, and in the context of the present deliverable, the design and technical specification of the GOV module continue to evolve, with significant architectural improvements implemented since the previous release. The concrete implementation has been substantially enhanced with new integration capabilities and automated processing features as detailed in the Module Demonstration section (see Section 5).

4.1 Integration into FSA

The GOV module is classified as a *container* named Operational Governance within the Transactional Operations *system*. The cooperation relationships identified there are with the Authentication and Authorization module and with the Provenance and Tracing module (previously Assets Trading & Monetization module). The integration now also includes connections to external applications (such as the FFGA) and API support for Payment Service Provider services for Digital Euro processing. There is also a mention of Service Level Agreements (SLA) as a GOV internal concern. It is worth noting, however, that the diagram depicted here comes from deliverable D2.3, which was released at an earlier stage of the project. Since then, due to the agile nature of our development process, cross-module relationships did undergo some revisions: for an up-to-date version of these, please refer to the *C4 container-level diagram* of the GOV module provided in Section 3.2 as Figure 3. Additionally, SLA management has been moved under the scope of task T4.1, as the definition of a proposed SLA -- if any is provided by an asset publisher -- is now done as part of the *offering definition* (see deliverable D4.1), while its enforcement is not in the scope of the FAME Platform at all.

Key Architectural Changes:

- External Application Support: The module now provides dedicated internal APIs for integration with external governance applications

- Payment Service Provider Integration: API support for Digital Euro processing (implementation pending)
- P&T Module Integration: Blockchain operations are now handled through the Provenance & Tracing module
- Enhanced Security Model: Improved separation between public and internal APIs with enhanced access controls

The goal of the *Operational Governance* platform module is to provide the necessary services for the management and the day-to-day operation of the platform. These services are exposed by four components: Federation Management, User Management, User Support, and Trading Support.

Architectural Enhancements: the component architecture has been enhanced to support both traditional direct API usage and integration with external applications, providing flexibility for different deployment scenarios and user interfaces. The blockchain integration layer has been abstracted through the P&T module, while payment processing APIs have been prepared for future PSP integration.

4.2 Workflows and Integration with Other Modules

The GOV workflows that are worth mentioning are those that enable the following objectives:

- Onboarding and Offboarding of Federation Member (enhanced with external application support and offboarding capabilities)
- Onboarding and Offboarding of User (enhanced with external application support and offboarding capabilities)
- Enrollment and Disenrollment of Trading Account (enhanced with P&T module integration)
- Procurement and Redemption of FDE coins (enhanced with API support for automated PSP integration)

For each of these, GOV also supports a workflow that operates in the opposite direction - e.g., "Offboarding of Federation Member", "Offboarding of User", "Redemption of FDE coins" - and these reverse workflows are now fully implemented and documented.

In the step-by-step description of workflows that follow, some steps are performed with tools that are external to the FAME Platform. Good examples are the interactions of users with Digital Euro payment systems (as in Section 3.3.4) and the onboarding of users made by external Onboarding Authorities (see Section 3.3.2.1). New external interactions include the FAME Federation Governance Application (FFGA) for member and user management. These steps can be easily spotted as *their descriptive text is in italics*.

The workflows are described assuming that all actors will interact with the GOV module through the FAME Dashboard or through external applications such as the FFGA, and that user authentication will be performed at the Dashboard level by means of the FAME Authentication & Authorization Infrastructure (AAI). With respect to the latter topic, GOV's Open API service endpoints work in the same way as all public services of the FAME platform: the security context, which has the form of a FAME-issued *token* and ensures that the caller's attributes are trustworthy, is established by the login process *in the web browser of the user*; it is then retrieved by the Dashboard web page or external application and finally propagated to the Open API service endpoint. Access control decisions (i.e., authorization and/or content filtering) are made, case by case, by the web page or external application and *repeated* by the service endpoint (to ensure that direct calls will not bypass any control). Although the details of this process are not relevant for this deliverable, a high-level picture of it is provided here for better understanding.

```
┌─────────────────────────────────────────────────────────────────┐
│ 1. User Browser Access                                          │
│    (User → Web Browser → FAME Dashboard/External App)           │
└─────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼───────────────────────────────────────────┐
│ 2. Login Initiation                                             │
│    (Dashboard/External App → FAME AAI → Displays QR Code)       │
└─────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼───────────────────────────────────────────┐
│ 3. VOC Presentation                                             │
│    (User → Scans QR Code with Identity Wallet app)              │
└─────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼───────────────────────────────────────────┐
│ 4. OID4VP Protocol                                              │
│    (Identity Wallet ↔ FAME AAI: VOC verification via OID4VP)    │
└─────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼───────────────────────────────────────────┐
│ 5. Entity Validation & Blacklist Check (NEW)                    │
│    ┌──────────────────────────────────────────────────────────┐ │
│    │ AAI → GOV Module Internal API:                           │ │
│    │ • Verify issuer OA is valid and not blacklisted          │ │
│    │ • Verify affiliation member is valid and not blacklisted │ │
│    │ • Verify user is valid and not blacklisted               │ │
│    │ • Confirm all entities have active status                │ │
│    └──────────────────────────────────────────────────────────┘ │
└─────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼───────────────────────────────────────────┐
│ 6. JWT Token Issuance                                           │
│    (FAME AAI → Issues JWT bearer token to user's browser)       │
└─────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼───────────────────────────────────────────┐
│ 7. Token Retrieval                                              │
│    (Dashboard/External App → Retrieves JWT token from browser)  │
└─────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼───────────────────────────────────────────┐
│ 8. API Call with Token                                          │
│    (Dashboard/External App → GOV API with JWT token)            │
└─────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼───────────────────────────────────────────┐
│ 9. Authorization Decision                                       │
│    ┌──────────────────────────────────────────────────────────┐ │
│    │ • Dashboard/External App: Initial access control         │ │
│    │ • GOV API Endpoint: Repeated authorization check         │ │
│    │ • Prevents bypass of frontend controls                   │ │
│    │ • JWT token validates user's onboarded status            │ │
│    └──────────────────────────────────────────────────────────┘ │
└───────────────────────────────────────────────────────────────┘ │
                      │
┌─────────────────────▼───────────────────────────────────────────┐
│ 10. Service Execution                                           │
│     (Authorized request processed by GOV module)                │
└─────────────────────────────────────────────────────────────────┘
```

Figure 4 Workflows and Integration with Other Modules

4.2.1 Onboarding of Federation Member

This process happens mostly offline, the only involvement of the FAME Platform being the last step. From the FAME Platform's perspective, its purpose is to add a Federation Member information entity to the relevant registry, so that VOCs issued by the new member are accepted by FAME's AAI.

Enhanced Process: The onboarding process now supports both traditional administrative workflows and integration with external governance applications such as the FFGA.

4.2.1.1 Traditional Administrative Workflow

1. The candidate organization goes through an offline process to have its membership approved by the FAME federation governance board (FGB). As part of this process, its legal identity and contact information are communicated to the FGB and verified. If the new member must operate as an Onboarding Authority, the process will also require the member to set up a suitable VOC issuance system, and the FGB to assess its compatibility with FAME's specifications.

2. The admin user receives instructions from the FGB to proceed with the onboarding of the new member. The instructions include the new members' contact information. If the member must operate as an Onboarding Authority, the admin user will obtain from it the OA's Decentralized Identifier (DID).

3. The admin user connects to the FAME Admin Dashboard, going through the authentication & authorization process.

4. The admin user navigates to the Members section and registers the newly approved member as a Federation Member entry in the platform's registry.

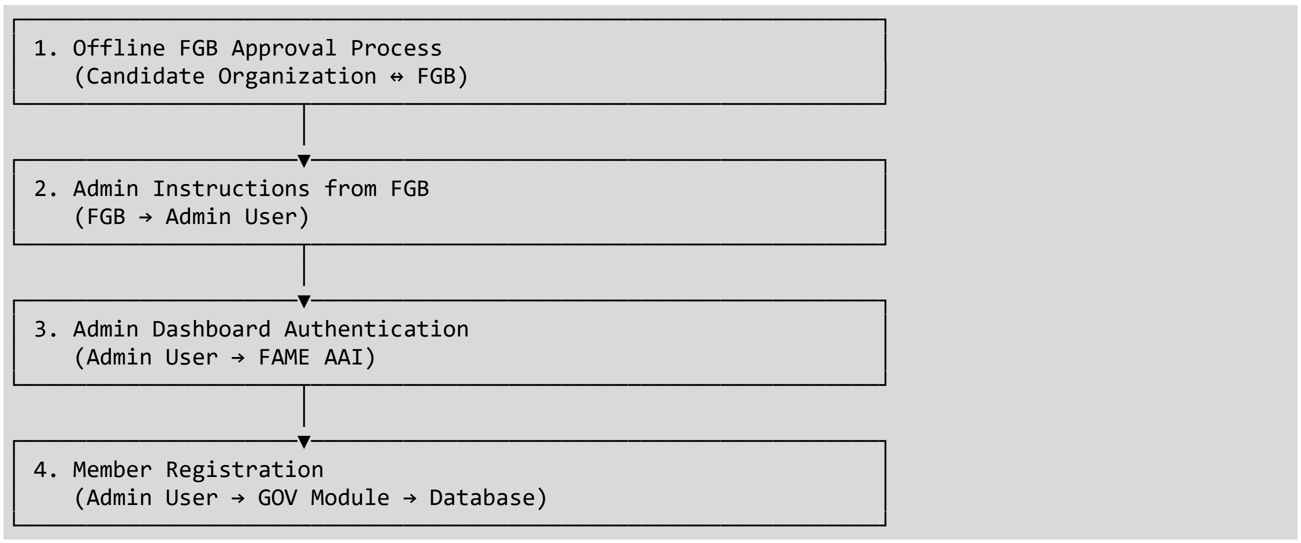

1. Offline FGB Approval Process
(Candidate Organization ↔ FGB)
2. Admin Instructions from FGB
(FGB → Admin User)
3. Admin Dashboard Authentication
(Admin User → FAME AAI)
4. Member Registration
(Admin User → GOV Module → Database)

Figure 5 Traditional Administrative Workflow

4.2.1.2 External Application Integration Workflow

New capability: The process can now be initiated and managed via external governance applications:

1. The candidate organization accesses the FAME Federation Governance Application (FFGA) and initiates the membership application process.

2. The FFGA guides the organization through the required information collection, including legal entity details, contact information, and organizational documentation.

3. The FFGA submits the membership application to the GOV module through the internal member onboarding API (see Section 4.2.2.2).

4. The GOV module processes the application and returns a tracking identifier for status monitoring.

5. The FFGA provides real-time status updates to the candidate organization and facilitates any required additional information collection.

6. Upon approval, the member registration is completed automatically through the integrated workflow.

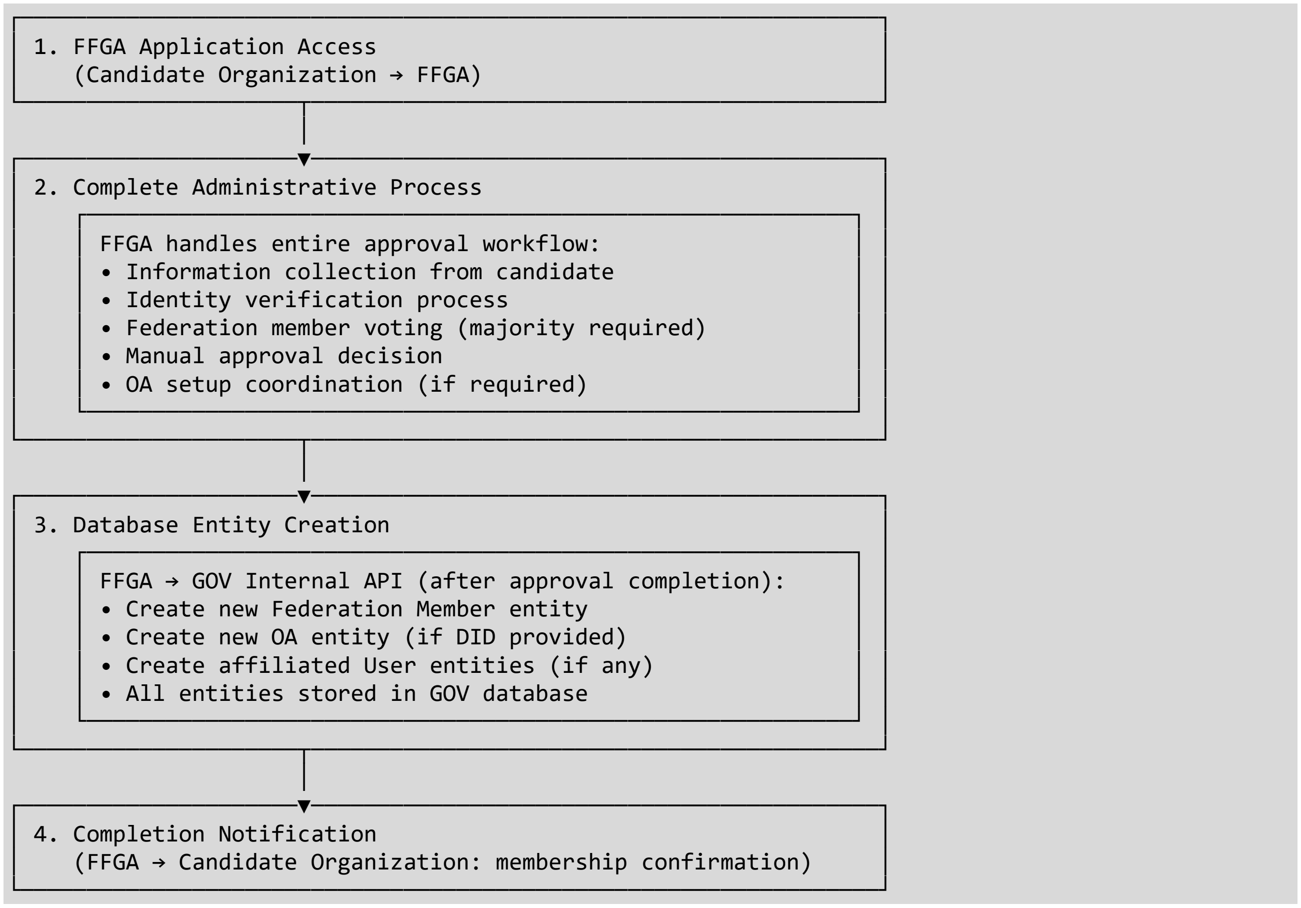

1. FFGA Application Access
(Candidate Organization → FFGA)
2. Complete Administrative Process
FFGA handles entire approval workflow:
• Information collection from candidate
• Identity verification process
• Federation member voting (majority required)
• Manual approval decision
• OA setup coordination (if required)
3. Database Entity Creation
FFGA → GOV Internal API (after approval completion):
• Create new Federation Member entity
• Create new OA entity (if DID provided)
• Create affiliated User entities (if any)
• All entities stored in GOV database
4. Completion Notification
(FFGA → Candidate Organization: membership confirmation)

Figure 6 External Application Integration Workflow

4.2.2 Offboarding of Federation Member

New Capability: The GOV module now supports the complete offboarding of federation members through administrative action.

1. An administrator identifies the need to offboard a federation member (due to policy violations, membership termination, regulatory requirements, etc.).
2. The administrator connects to the FAME Admin Dashboard, going through the authentication & authorization process.
3. The administrator navigates to the Members section and initiates the offboarding process for the target member.
4. **Cascading Deactivation Process**: The system automatically implements the following actions:
   - Updates the member status to TERMINATED
   - Deactivates all Trading Accounts owned by users affiliated with the offboarded member
   - Flags all user records affiliated with the member (preserving data for audit purposes)
   - Revokes all active permissions and authorizations
   - Notifies the P&T module to update blockchain permissions accordingly
   - Generates comprehensive audit trails for compliance
5. The offboarding is completed while preserving all data integrity for audit and compliance requirements.

Note: Member offboarding results in immediate cessation of all platform capabilities for the member and its affiliated users, while maintaining complete audit trails for regulatory compliance.

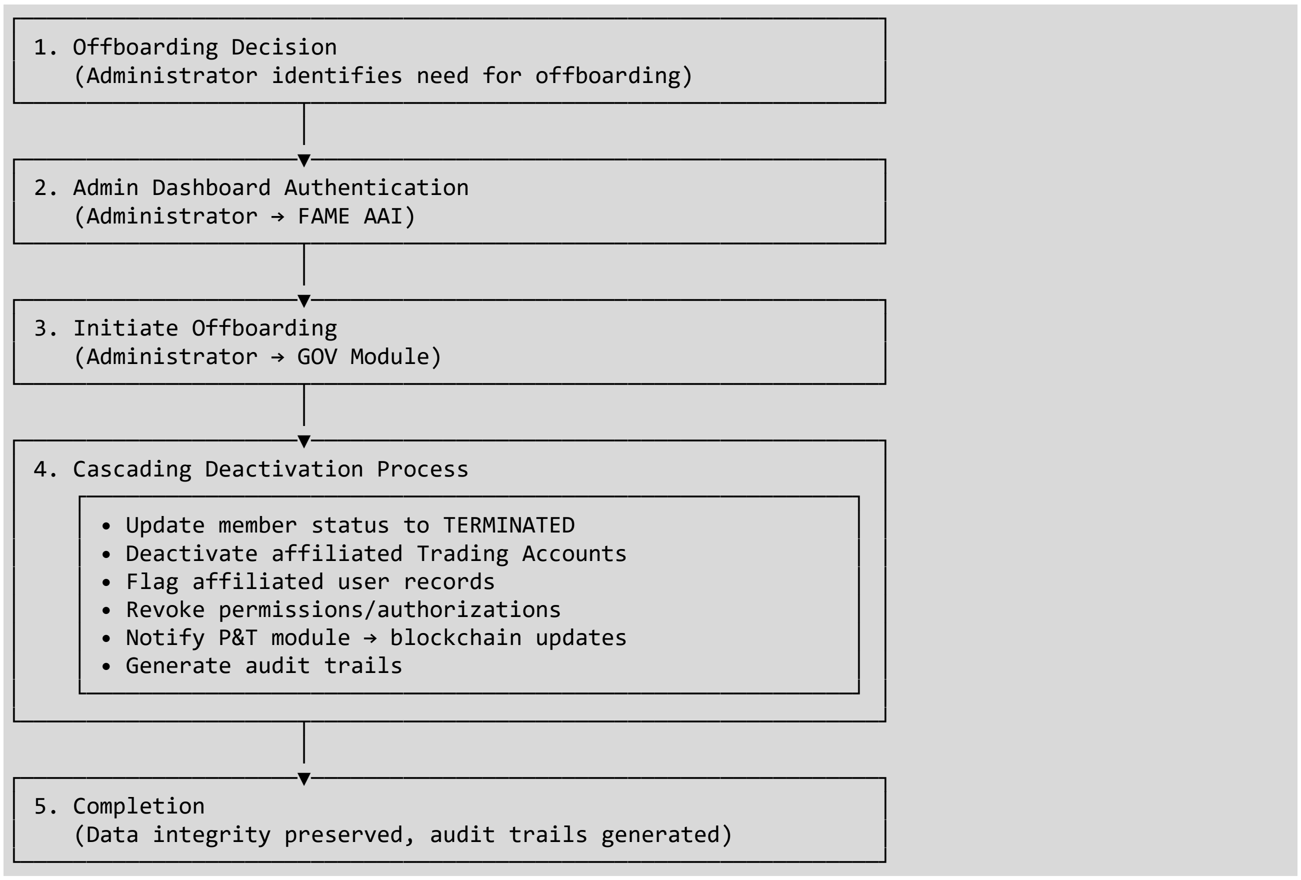

1. Offboarding Decision
(Administrator identifies need for offboarding)
2. Admin Dashboard Authentication
(Administrator → FAME AAI)
3. Initiate Offboarding
(Administrator → GOV Module)
4. Cascading Deactivation Process
• Update member status to TERMINATED
• Deactivate affiliated Trading Accounts
• Flag affiliated user records
• Revoke permissions/authorizations
• Notify P&T module → blockchain updates
• Generate audit trails
5. Completion
(Data integrity preserved, audit trails generated)

Figure 7 Workflow Offboarding of Federation Member

4.2.3 Onboarding of User

There are two similar but different processes for the onboarding of users:

- **Regular users** are people who participate in the FAME data marketplace as traders or guests: they are onboarded by an Onboarding Authority (OA), which is a legal entity that is member of the FAME federation.
- **Platform operators** are people having administrative tasks on the FAME Platform: they are onboarded by a Root Authority (RA), which is an organization that is *a core member* of the FAME federation (typically, this means that it directly contributes to hosting/running the IT infrastructure that underpins the FAME Platform).

Regardless of being an OA or RA, such role implies some responsibilities:

- Verification of user's identity.
- Management of any personal data the authority is in possession of.
- Compliance with all regulation that applies to such personal data (e.g., GDPR).

To avoid placing the big burden of GDPR compliance - and associated risks and costs - on the core members of the FAME federation, the onboarding and management of the *vast majority* of users is delegated to the OAs. Moreover, FAME's architecture is designed so that no personally identifiable information (PII) is transferred from the OA to the platform. On the other hand, a RA is only responsible for a handful of users: platform operators, *who will typically already have employee status*.

From the GOV module's perspective, these two scenarios are very different. The first is not supported at all, because all onboarding activities are performed on the OA's side, resulting into a VOC that carries no PII and that will only be checked for validity when presented by the user (in other words: the process is a concern of the AAI module). The second is supported in the simplest of ways: platform operators are registered with a minimum of contact information, and VOC issuance is a straightforward online workflow that requires users to follow an invitation link to a web page and scan a QR-Code with their FAME Identity Wallet app.

4.2.3.1 Onboarding of Regular User

1. The candidate is known to the OA and goes through an internal approval process. If no identity verification was done previously, it [must]{.underline} be included in the process.

2. The candidate, now an approved but not yet onboarded FAME user, obtains the FAME Identity Wallet app - or any other OID4VC-compatible app - and installs it on their personal mobile device.

3. The user, through any OA-specific process, retrieves their VOC and stores it in their wallet app.

4.2.3.2 Onboarding of Platform Operator

Enhanced Process: Platform operator onboarding now supports both traditional administrative workflows and external application integration.

4.2.3.3 Traditional Administrative Workflow

1. The candidate, who is an endorsed but not yet onboarded FAME platform operator, obtains the FAME Identity Wallet app and installs it on their personal mobile device.

2. The admin user connects to the FAME Admin Dashboard, going through the authentication & authorization process.

3. The admin user navigates to the Operators section and registers a new User entity, inserting the required contact information and setting the assigned role.

4. The system sends an invitation message to the email address - and, if provided, to the phone number - included in User entity. The message contains a link to a dedicated Dashboard page, together with an *invitation ID* that is specific to the current onboarding workflow.

5. The invited user follows the received link, opening the Dashboard page in their web browser. The page prompts the user to submit a one-time-password (OTP) in order to proceed further.
6. The system, having detected the user and checked the status of the pending invitation, generates a workflow-specific OTP and sends it to the user's contact point(s) - see step #3.

7. The invited user submits the received OTP through the Dashboard page.

8. The system verifies that the OTP is correct and issues the user's VOC. It then displays a QR-Code that encodes the information needed for the retrieval of the VOC from the platform.

9. The invited user activates their wallet app and scans the QR-Code.

10. The wallet app, after asking confirmation from the user, retrieves the VOC from the platform and stores it locally.

```
┌─────────────────────────────────────────────┐
│ 1. Wallet App Installation                  │
│    (Candidate → Downloads FAME Identity Wallet) │
└──────────────────────┬──────────────────────┘
                       │
┌──────────────────────▼──────────────────────┐
```

```
│ 2. Admin Dashboard Authentication                                   │
│    (Admin User → FAME AAI)                                          │
└─────────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼───────────────────────────────────────────────┐
│ 3. User Registration                                                │
│    (Admin User → GOV Module → Create User entity with role)         │
└─────────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼───────────────────────────────────────────────┐
│ 4. Invitation Message                                               │
│    (System → Email/SMS with invitation link & ID)                   │
└─────────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼───────────────────────────────────────────────┐
│ 5. Dashboard Access                                                 │
│    (User → Follows link → Dashboard page → OTP prompt)              │
└─────────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼───────────────────────────────────────────────┐
│ 6. OTP Generation & Delivery                                        │
│    (System → Generates workflow-specific OTP → Email/SMS)           │
└─────────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼───────────────────────────────────────────────┐
│ 7. OTP Submission                                                   │
│    (User → Submits OTP via Dashboard)                               │
└─────────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼───────────────────────────────────────────────┐
│ 8. VOC Issuance & QR Code                                           │
│    (System → Verifies OTP → Issues VOC → Displays QR code)          │
└─────────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼───────────────────────────────────────────────┐
│ 9. Wallet Integration                                               │
│    (User → Scans QR code with wallet app)                           │
└─────────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼───────────────────────────────────────────────┐
│ 10. VOC Storage                                                     │
│     (Wallet App → Retrieves & stores VOC locally)                   │
└─────────────────────────────────────────────────────────────────────┘
```

Figure 8 Traditional Administrative Workflow

4.2.3.4 External Application Integration Workflow

New capability: Platform operator onboarding can now be initiated through external applications:

1. The candidate accesses an external governance application (such as FFGA) and initiates the platform operator registration process.

2. The external application collects the required user information and role assignments through its interface.

3. The external application submits the onboarding request to the GOV module through the internal user onboarding API (see Section 4.3.2.2).

4. The GOV module processes the request and manages the invitation workflow automatically, including OTP generation and VOC issuance coordination.

5. The external application provides status updates and user guidance throughout the credential acquisition process.

```
┌─────────────────────────────────────────────────────────────────────┐
```

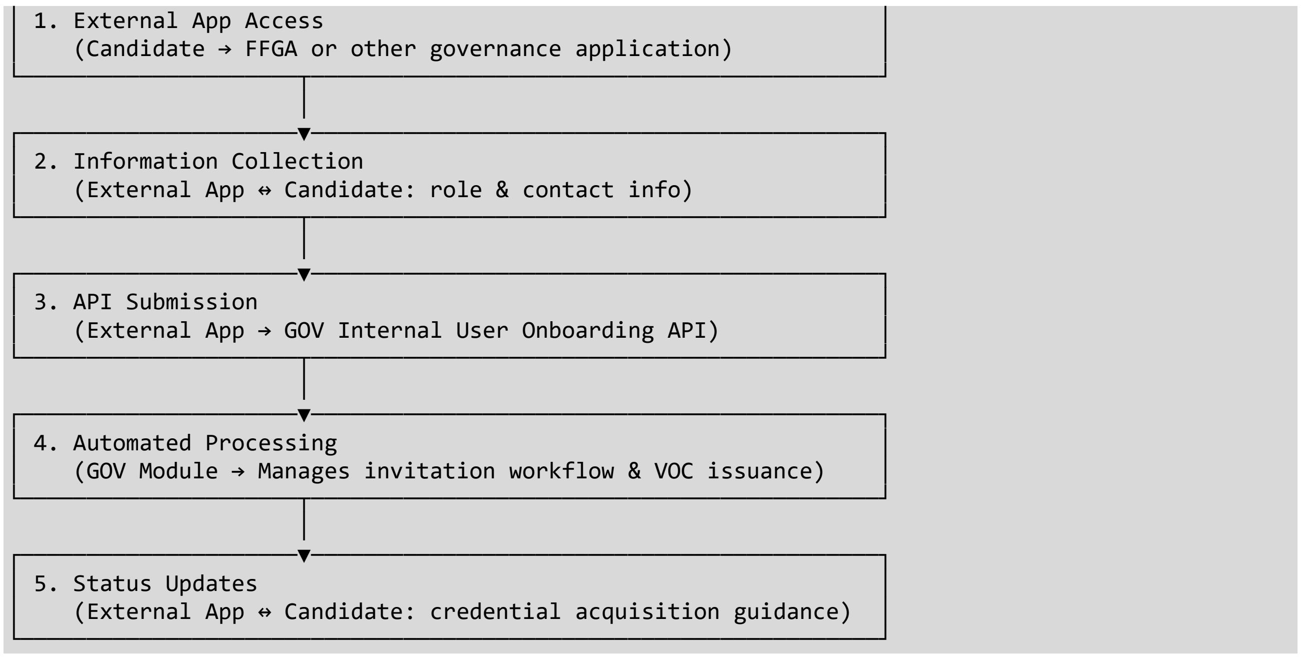

1. External App Access
(Candidate → FFGA or other governance application)
2. Information Collection
(External App ↔ Candidate: role & contact info)
3. API Submission
(External App → GOV Internal User Onboarding API)
4. Automated Processing
(GOV Module → Manages invitation workflow & VOC issuance)
5. Status Updates
(External App ↔ Candidate: credential acquisition guidance)

Figure 9 External Application Integration Workflow

4.2.4 Offboarding of User

New Capability: The GOV module now supports comprehensive user offboarding through administrative action.

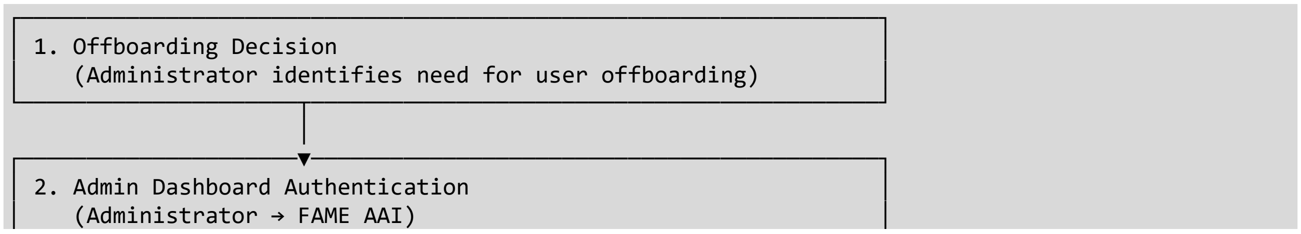
1. An administrator identifies the need to offboard a user (role changes, security concerns, policy violations, etc.).

2. The administrator connects to the FAME Admin Dashboard, going through the authentication & authorization process.

3. The administrator navigates to the Users section and initiates the offboarding process for the target user.

4. **Immediate Access Revocation**: The system implements the following actions:
   - Updates the user status to OFFBOARDED
   - Revokes all active authentication tokens
   - Adds the user to the authentication blacklist
   - Deactivates all Trading Accounts owned by the user
   - Notifies the P&T module to revoke blockchain permissions
   - Initiates VOC revocation process (when fully implemented)
   - Generates comprehensive audit trails
5. The user is immediately unable to authenticate with the platform while all data is preserved for audit purposes.

Note: User offboarding results in immediate loss of all platform access while maintaining complete audit trails for compliance and security monitoring.


1. Offboarding Decision
(Administrator identifies need for user offboarding)
2. Admin Dashboard Authentication
(Administrator → FAME AAI)

```
└─────────────────────────────────────────────────────────────────┘
                      │
                      ▼
┌─────────────────────────────────────────────────────────────────┐
│ 3. Initiate User Offboarding                                    │
│    (Administrator → GOV Module → User Management)               │
└─────────────────────────────────────────────────────────────────┘
                      │
                      ▼
┌─────────────────────────────────────────────────────────────────┐
│ 4. Immediate Access Revocation                                  │
│     ┌───────────────────────────────────────────────────────────┐ │
│     │ • Update user status to OFFBOARDED                        │ │
│     │ • Revoke all authentication tokens                        │ │
│     │ • Add user to authentication blacklist                    │ │
│     │ • Deactivate user's Trading Accounts                      │ │
│     │ • Notify P&T module → revoke blockchain permissions       │ │
│     │ • Initiate VOC revocation (when implemented)              │ │
│     │ • Generate comprehensive audit trails                     │ │
│     └───────────────────────────────────────────────────────────┘ │
└─────────────────────────────────────────────────────────────────┘
                      │
                      ▼
┌─────────────────────────────────────────────────────────────────┐
│ 5. Completion                                                   │
│    (User unable to authenticate, audit trails preserved)        │
└─────────────────────────────────────────────────────────────────┘
```

Figure 10 Offboarding of User Workflow

4.2.5 Enrollment of Trading Account

Trading Accounts are a staple of FAME's data marketplace: they ensure that all the business models supported in FAME (see Section 2.6) can work by means of *online transactions*, where access to data assets is unlocked by a real-time payment. A Trading Account is a FAME *extension* to a plain blockchain account. It is based on a regular, Ethereum-style account but has two additional features:

- **Permissioning** - The FAME Blockchain Infrastructure is a *permissioned* network, which means that blockchain accounts must be included in a system-managed *allowlist* to be authorized to execute any smart contract, or to submit blockchain transactions in general.
  Pseudonymous ownership - FAME blockchain accounts are not entirely anonymous as in the Ethereum world: they are mapped to the ID of the user who owns them - basically, to a pseudonym that, in case of need, can be traced back to a real identity with the cooperation of the OA who originally onboarded the user.

Both features are enabled by one single workflow: Trading Account enrollment. This step is mandatory for any user who, having already gone through onboarding, wants to either buy or sell assets on the FAME data marketplace.

Significant Architectural Change: The enrollment process has been enhanced to integrate with the Provenance and Tracing (P&T) module rather than managing blockchain permissions directly.

1. The user, by means of any suitable blockchain wallet software, creates an Ethereum-style blockchain account and copies down the resulting Ethereum address.

2. The user connects to the FAME End-user Dashboard, going through the authentication & authorization process. This process ensures that the incoming user is associated with the correct user ID (UID) and, if present, with the provenance ID (PID) that indicates the user's affiliation.

3. The user navigates to the Profile section, where a web form is displayed that prompts the user to insert a Trading Account ID (TID) for enrollment.

4. The user inserts the blockchain account address as the TID value and submits the form. The user's UID is also included in the submission.

5. The system checks that the given TID is valid and that is not already enrolled as a Trading Account. It then turns the enrollment request into a User Request message, which is enqueued for being processed in the background. The request ID is immediately returned to the Dashboard page, where it is used as a handle for monitoring the progress of the process.

6. Enhanced Integration: While the user is waiting for status updates, the system submits the enrollment request to the P&T module through its internal API rather than directly interacting with the blockchain. The P&T module handles all blockchain-level operations including allowlist management.

7. P&T Module Processing: The P&T module validates the account, adds it to the blockchain allowlist, and confirms the operation success or failure back to the GOV module through a callback API.

8. When the blockchain transaction is confirmed by the P&T module, the GOV module updates the Trading Account database, adding a mapping between the TID and the UID/PID of the requesting user. It also updates the User Request message in the request queue, marking the process as successfully completed.

9. The user, still on the same page from which the enrollment request was filed, checks that their request was successfully processed.

Benefits of P&T Integration:

- Separation of Concerns: GOV focuses on governance while P&T handles blockchain operations
- Enhanced Reliability: Specialized blockchain interaction handling with improved error recovery
- Centralized Blockchain Management: Consistent blockchain permission management across the platform
- Improved Audit Trails: Comprehensive logging of all blockchain-related operations

```
┌─────────────────────────────────────────────────────────────────┐
│ 1. Blockchain Account Creation                                  │
│    (User creates Ethereum account via wallet software)          │
└─────────────────────────────────────────────────────────────────┘
                              │
┌─────────────────────────────▼───────────────────────────────────┐
│ 2. Dashboard Authentication                                     │
│    (User → FAME AAI, establishes UID/PID context)               │
└─────────────────────────────────────────────────────────────────┘
                              │
┌─────────────────────────────▼───────────────────────────────────┐
│ 3. Account Enrollment Request                                   │
│    (User → Dashboard → TID submission)                          │
└─────────────────────────────────────────────────────────────────┘
                              │
┌─────────────────────────────▼───────────────────────────────────┐
│ 4. Request Validation & Queuing                                 │
│    (GOV Module validates TID, creates User Request)             │
└─────────────────────────────────────────────────────────────────┘
                              │
┌─────────────────────────────▼───────────────────────────────────┐
│ 5. P&T Module Integration (UPDATED)                             │
│    (GOV → P&T Internal API instead of direct blockchain)        │
└─────────────────────────────────────────────────────────────────┘
                              │
┌─────────────────────────────▼───────────────────────────────────┐
│ 6. Blockchain Processing                                        │
│    (P&T Module → Blockchain allowlist management)               │
```

```
└────────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌────────────────────────────────────────────────────────────────┐
│ 7. Confirmation Callback                                       │
│    (P&T Module → GOV Module via callback API)                  │
└────────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌────────────────────────────────────────────────────────────────┐
│ 8. Database Update & Request Completion                        │
│    (GOV updates Trading Account DB, marks User Request complete)
└────────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌────────────────────────────────────────────────────────────────┐
│ 9. User Confirmation                                           │
│    (User checks enrollment status via Dashboard)               │
└────────────────────────────────────────────────────────────────┘
```

Figure 11 Enrollment of Trading Account

4.2.6 Procurement of FDE coins

The main function of Trading Accounts is to send and receive payment tokens, that in FAME are implemented as a digital currency named "FDE coin" . As FDE coins are a digital representation of the EUR currency (a pattern known as *stablecoin*), and the FAME ecosystem is not supposed to create actual money out of thin air, the existence of every single FDE coin is connected with that of a matching EUR deposited at an *escrow account*. In practice: users get FDE coins in their Trading Account - and can thus make payments in the FAME data marketplace - on condition of having sent the equivalent amount of EUR to a special, FAME-managed bank account; they also get EURs in *their* bank account on condition of having the matching amount of FDEs removed (and *burnt* by the system for good) from their Trading Account. The former workflow is called **procurement**, the latter, **redemption**.

Important Note: The procurement and redemption workflows have been enhanced with API support for integration with a Payment Service Provider (PSP) that supports Digital Euro operations. However, this integration is currently implemented at the API level only and is not yet deployed in the integrated platform. The system currently maintains the traditional manual process while providing the foundation for future automated operations.

The procurement workflow is unraveled below. The prerequisite for both has been updated: the system now includes API support for integration with the European Central Bank's Digital Euro infrastructure through a certified PSP, though this remains unimplemented in the current deployment.

4.2.6.1 Enhanced Procurement Workflow (API Ready)

The procurement workflow has been prepared for automated operation through PSP integration:

1. The user connects to the FAME End-user Dashboard, going through the authentication & authorization process.
2. The user navigates to the Profile section, and from there to the page that displays the details and status of a Trading Account owned by the user.
3. The user starts the "request funds" process, to the effect that the page displays a web form for submitting a request for receiving funds on the currently displayed Trading Account.
4. The user inputs the amount of requested FDE coins and submits the web form.
5. **PSP Integration (API Ready)**: The system includes support for automatic redirection to an integrated PSP interface for Digital Euro authorization and payment processing, though this integration is not yet activated in the current deployment.

6. **Traditional Process (Current Implementation)**: The system turns the request into a Support Ticket, which is created in "pending" state and contains the details of the funding request.
7. The operator connects to the FAME Admin Dashboard, going through the authentication & authorization process.
8. The operator navigates to the Helpdesk page, where the pending Support Ticket is listed, and takes on responsibility for it (i.e., the operator becomes the ticket's *assignee*).

NOTE: From now on, all communication between the user and the operator happens through the channel represented by the Support Ticket abstraction.

9. The user support operator sends the escrow account coordinates to the user.
10. The user transfers EUR-denominated funds to the escrow account. The transferred amount matches the funding request.
11. The operator verifies that the escrow account did receive the correct amount of EUR-denominated funds.
12. The operator navigates to the Traders section of the Admin Dashboard, and from there to the page that displays the details and status of the Trading Account for which the funding request was submitted.
13. The operator starts the "manage tokens" process, to the effect that the page displays a web form for minting and crediting FDE tokens to the currently displayed Trading Account.
14. The operator inputs the amount of FDE coins to mint and credit, and submits the web form.
15. The system submits the token minting request to the P&T module through its internal API for blockchain processing.
16. In due time, the operator verifies, on the Trading Account detail page (see step #12), that the balance has the expected value. The operator then communicates to the user the completion of their support request and closes the Support Ticket.
17. The user, at any time, can verify that the FDE funds have been received by checking the Trading Account balance from its detail page (see step #2).

```
┌────────────────────────────────────────────────────────────────┐
│ 1. Dashboard Access                                            │
│    (User → FAME Dashboard → Trading Account page)              │
└────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼──────────────────────────────────────────┐
│ 2. Fund Request                                                │
│    (User submits funding request form)                         │
└────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼──────────────────────────────────────────┐
│ 3. Support Ticket Creation                                     │
│    (System creates pending Support Ticket)                     │
└────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼──────────────────────────────────────────┐
│ 4. Operator Assignment                                         │
│    (Operator → Admin Dashboard → takes ticket responsibility)  │
└────────────────────────────────────────────────────────────────┘
                      │
┌─────────────────────▼──────────────────────────────────────────┐
│ 5. Banking Coordination                                        │
│   (Operator ↔ User: escrow account details)                     │
└─────────────────────────────────────────────────────────────────┘
                      │
                      ▼
┌─────────────────────────────────────────────────────────────────┐
│ 6. EUR Transfer                                                 │
│    (User → Traditional Banking → Escrow Account)                │
└─────────────────────────────────────────────────────────────────┘
                      │
                      ▼
┌─────────────────────────────────────────────────────────────────┐
│ 7. Payment Verification                                         │
│    (Operator verifies EUR receipt)                              │
└─────────────────────────────────────────────────────────────────┘
                      │
                      ▼
┌─────────────────────────────────────────────────────────────────┐
│ 8. Token Minting Request                                        │
│    (Operator → GOV Module → P&T Module API)                     │
└─────────────────────────────────────────────────────────────────┘
                      │
                      ▼
┌─────────────────────────────────────────────────────────────────┐
│ 9. Blockchain Minting                                           │
│    (P&T Module → Blockchain → FDE coin creation)                │
└─────────────────────────────────────────────────────────────────┘
                      │
                      ▼
┌─────────────────────────────────────────────────────────────────┐
│ 10. Completion                                                  │
│     (Operator closes ticket, user verifies balance)             │
└─────────────────────────────────────────────────────────────────┘
```

Figure 12 Enhanced Procurement Workflow

5 Components Specification

This section contains the technical documentation of the Open API (user- and application-level service endpoints) and Integration API (cross-module integration points) of the GOV module, grouped by component.

Open API and Integration API operations can be distinguished by looking at the "API Visibility" paragraph of their documentation entry:

- **Open API operations** have PUBLIC visibility: they *must* be exposed over the public Internet using the secure version of the HTTP protocol, and their access *must* be mediated by the FAME Authentication and Authorization Infrastructure. To this goal, their visibility paragraph also describes the Access Control rules that are applied - e.g., "authenticated user with admin role".
- **Integration API operations** have INTERNAL visibility: they *must not* be exposed over the public Internet. It is assumed that the LAN used to connect the different modules of the FAME Platform is configured in such way that access to Integration API endpoints is only possible from local IP addresses.

All the API operations, regardless of component or visibility, follow the REST-over-HTTP paradigm.

Note on Implementation Status: This updated specification reflects significant changes and improvements made since the previous release. Some components have been entirely overhauled (notably Trading Support), while others remain documented but not yet fully implemented (User Support). New endpoints and enhanced functionality have been added throughout the system to better support the FAME marketplace operations.

5.1 Baseline Technologies and Tools

The baseline technologies and tools remain largely unchanged from the previous version. The only exception is the removal of blockchain technology from this module, as all interactions with the FAME Blockchain Infrastructure are now mediated by the P&T module, which provides an internal API for integration.

Baseline Technology	Description	Usage in FAME
Node.js	Runtime environment for server-side applications	Implementation of service endpoints
NestJS	Framework for server-side applications	Implementation of service endpoints
TypeScript language	JavaScript-style programming with object/data types	Implementation of service endpoints
MongoDB	No-SQL database platform	Persistence for User Request entities
MySQL	SQL database platform	Persistence for all other entities
Sphereon SSI SDK	Framework for self-sovereign identity applications	Implementation of OID4VCI protocol

Node.js	Runtime environment for server-side applications	Implementation of service endpoints
NestJS	Framework for server-side applications	Implementation of service endpoints
TypeScript language	JavaScript-style programming with object/data types	Implementation of service endpoints

Table 6 Baseline Technologies and Tools

5.2 Federation Management

5.2.1 Description

The Federation Management component provides the Open API for the administration of federation members: it supports their onboarding/offboarding, the management of their records, and implements a directory for quick membership checks. The definition of "federation member", in the FAME context, is the following: a *legal entity* that has been approved by the federation's governance body as a trusted contributor to the FAME's catalogue and/or as a trusted authority for the onboarding of FAME users.

Significant Updates: This component has been enhanced with improved member lifecycle management, including new migration capabilities for updating legal entity information and more comprehensive member status tracking. The API now provides better separation between public and internal operations, with enhanced security controls.

Authorities fall into two categories:

- **Onboarding Authorities (OA)** are entities that are entrusted with the onboarding of "regular" users - i.e., publishers and consumers of data assets
- **Root Authorities (RA)** are a special case of OA, as they can onboard users with administrative privileges

The component is, at its core, a comprehensive registry with advanced CRUD capabilities, enhanced validation, and improved integration with other platform modules.

5.2.2 Technical Specification

5.2.2.1 Public API Interfaces

Onboard Member — Starts the process of onboarding a new member

API Visibility: PUBLIC - Access Control: authenticated user with admin role

POST `/api/v1.0/members`

Request Body (Content-Type: application/json):

```
{
  "org": {
    "legalName": "string",
    "legalAddress": {
      "addressLine1": "string",
      "addressLine2": "string",
      "addressLine3": "string",
      "city": "string",
      "region": "string",
      "postCode": "string",
      "country": "string"
    },
    "lei": "string",
    "duns": "string",
    "bic": "string",
    "tin": "string"
  },
  "type": "FPO|NPO|RES|EDU|LEO|PUB|UKN",
  "rep": {
    "firstName": "string",
    "lastName": "string",
    "email": "string",
    "phone": "string"
  },
  "oa": "string"
}
```

Response Body (Content-Type: application/json):

```
{
  "id": "string"
}
```

Response Codes:

- 202: The request has been accepted and will be processed offline
- 400: Bad Request: missing or invalid input data
- 401: Unauthorized: missing or invalid authentication
- 403: Forbidden: access denied due to policy
- 500: Internal Server Error: something went wrong, the operation failed
- 502: Bad Gateway: an upstream server was unable to process this request

---

Update Member — Updates the non-trusted information linked to a member

API Visibility: PUBLIC - Access Control: authenticated user with admin role

PUT `/api/v1.0/members/{pid}`

Path Parameters:

- `pid` - Member PID

Request Body (Content-Type: application/json):

```
{
  "type": "FPO|NPO|RES|EDU|LEO|PUB|UKN",
  "rep": {
    "firstName": "string",
    "lastName": "string",
    "email": "string",
    "phone": "string"
  },
  "oa": "string"
}
```

Response Codes:

- 204: The operation was completed successfully
- 400: Bad Request: missing or invalid input data
- 401: Unauthorized: missing or invalid authentication
- 403: Forbidden: access denied due to policy
- 404: Not Found: the target resource does not exist or is in invalid state
- 500: Internal Server Error: something went wrong, the operation failed
- 502: Bad Gateway: an upstream server was unable to process this request

---

Migrate Member — Starts the process of changing the legal entity information of an existing member

API Visibility: PUBLIC - Access Control: authenticated user with admin role

POST `/api/v1.0/members/{pid}/migrate`

Path Parameters:

- `pid` - Member PID

Request Body (Content-Type: application/json):

```
{
  "legalName": "string",
  "legalAddress": {
    "addressLine1": "string",
    "addressLine2": "string",
    "addressLine3": "string",
    "city": "string",
    "region": "string",
    "postCode": "string",
    "country": "string"
  },
  "lei": "string",
  "duns": "string",
  "bic": "string",
  "tin": "string"
}
```

Response Body (Content-Type: application/json):

```
{
  "id": "string"
}
```

Response Codes:

- 202: The request has been accepted and will be processed offline
- 400: Bad Request: missing or invalid input data
- 401: Unauthorized: missing or invalid authentication

- 403: Forbidden: access denied due to policy
- 404: Not Found: the target resource does not exist or is in invalid state
- 500: Internal Server Error: something went wrong, the operation failed
- 502: Bad Gateway: an upstream server was unable to process this request

---

Offboard Member — Starts the process of offboarding an existing member

API Visibility: PUBLIC - Access Control: authenticated user with admin role

POST `/api/v1.0/members/{pid}/offboard`

Path Parameters:

- `pid` - Member PID

Response Codes:

- 202: The request has been accepted and will be processed offline
- 401: Unauthorized: missing or invalid authentication
- 403: Forbidden: access denied due to policy
- 404: Not Found: the target resource does not exist or is in invalid state
- 500: Internal Server Error: something went wrong, the operation failed
- 502: Bad Gateway: an upstream server was unable to process this request

---

List Members — List members

API Visibility: PUBLIC - Access Control: authenticated user with admin role

GET `/api/v1.0/members`

Query Parameters:

- `typ` - Organization type to filter members by (optional) Values: FPO, NPO, RES, EDU, LEO, PUB, UKN
- `cnt` - Country code to filter members by (optional)
- `sta` - Status to filter members by (optional) Values: ACTIVE, INACTIVE, TERMINATED

Response Body (Content-Type: application/json):

```
[
  {
    "pid": "string",
    "name": "string",
    "type": "FPO|NPO|RES|EDU|LEO|PUB|UKN",
    "country": "string",
    "active": boolean
  }
]
```

Response Codes:

- 200: Success
- 401: Unauthorized: missing or invalid authentication
- 403: Forbidden: access denied due to policy

- 500: Internal Server Error: something went wrong, the operation failed

---

Retrieve Member — Retrieve the full information set of a member

API Visibility: PUBLIC - Access Control: authenticated user with admin role

GET `/api/v1.0/members/{pid}`

Path Parameters:

- `pid` - PID of the member to retrieve

Response Body (Content-Type: application/json):

```
{
  "pid": "string",
  "org": {
    "legalName": "string",
    "legalAddress": {
      "addressLine1": "string",
      "addressLine2": "string",
      "addressLine3": "string",
      "city": "string",
      "region": "string",
      "postCode": "string",
      "country": "string"
    },
    "lei": "string",
    "duns": "string",
    "bic": "string",
    "tin": "string"
  },
  "type": "FPO|NPO|RES|EDU|LEO|PUB|UKN",
  "rep": {
    "firstName": "string",
    "lastName": "string",
    "email": "string",
    "phone": "string"
  },
  "authority": "string",
  "onboarded": "string",
  "offboarded": "string"
}
```

Response Codes:

- 200: Success
- 401: Unauthorized: missing or invalid authentication
- 403: Forbidden: access denied due to policy
- 404: Not Found: the target resource does not exist or is in invalid state
- 500: Internal Server Error: something went wrong, the operation failed
- 502: Bad Gateway: an upstream server was unable to process this request

---

Check Membership — Check active membership

API Visibility: PUBLIC - Access Control: authenticated user with admin role

GET `/api/v1.0/members/{pid}/check`

Path Parameters:

- `pid` - PID of the member to check

Response Codes:

- 204: Member is active
- 401: Unauthorized: missing or invalid authentication
- 403: Forbidden: access denied due to policy
- 404: Not Found: the target resource does not exist or is in invalid state
- 500: Internal Server Error: something went wrong, the operation failed

---

List Active Members — List active members

API Visibility: PUBLIC - Access Control: authenticated user

GET `/api/v1.0/active-members`

Response Body (Content-Type: application/json):

```
[
  {
    "pid": "string",
    "name": "string",
    "type": "FPO|NPO|RES|EDU|LEO|PUB|UKN",
    "country": "string",
    "active": boolean
  }
]
```

Response Codes:

- 200: The response body contains a list of references to active members
- 401: Unauthorized: missing or invalid authentication
- 500: Internal Server Error: something went wrong, the operation failed

---

Retrieve Active Member Reference — Resolve the PID of an active member

API Visibility: PUBLIC - Access Control: authenticated user

GET `/api/v1.0/active-members/{pid}`

Path Parameters:

- `pid` - PID to be resolved

Response Body (Content-Type: application/json):

```
{
  "pid": "string",
  "name": "string",
  "type": "FPO|NPO|RES|EDU|LEO|PUB|UKN",
  "country": "string",
  "active": boolean
}
```

Response Codes:

- 200: The response body contains the basic details of the matching member
- 401: Unauthorized: missing or invalid authentication
- 404: Not Found: the target resource does not exist or is in invalid state
- 500: Internal Server Error: something went wrong, the operation failed

5.2.2.2 Internal API Interfaces

Onboard Member (Internal) — Starts the onboarding of a new member

API Visibility: INTERNAL

POST `/gov/v1.0/members`

Request Body: Same as public API *Response Body*: Same as public API

Confirm Source Registration — Confirms source registration (callback operation)

API Visibility: INTERNAL

PUT `/gov/v1.0/members/{pid}/confirm-registration`

Path Parameters:

- `pid` - PID of the member being onboarded

Response Codes:

- 202: Success
- 400: Bad Request: missing or invalid input data
- 500: Internal Server Error: something went wrong, the operation failed
- 502: Bad Gateway: an upstream server was unable to process this request

---

Confirm Source Deregistration — Confirms source deregistration (callback operation)

API Visibility: INTERNAL

PUT `/gov/v1.0/members/{pid}/confirm-deregistration`

Path Parameters:

- `pid` - PID of the member being offboarded

Response Codes:

- 202: Success
- 400: Bad Request: missing or invalid input data
- 500: Internal Server Error: something went wrong, the operation failed
- 502: Bad Gateway: an upstream server was unable to process this request

---

Check Authority/Affiliation — Check authority / affiliation

API Visibility: INTERNAL

GET `/gov/v1.0/authorities/{did}/check`

Path Parameters:

- `did` - DID of the authority being checked

Query Parameters:

- `a` - PID of the affiliation being checked for consistency (optional)

Response Codes:

- 204: The combination of parameters is valid
- 404: Not Found: the target resource does not exist or is in invalid state
- 500: Internal Server Error: something went wrong, the operation failed

---

List Active Authorities — List active authorities

API Visibility: INTERNAL

GET `/gov/v1.0/active-authorities`

Response Body (Content-Type: application/json):

```
[
  {
    "did": "string",
    "pid": "string",
    "name": "string",
    "active": boolean
  }
]
```

5.2.2.3 Data Structures

Organization Types (MTYPE domain):

- FPO - For-Profit Organization
- NPO - Non-Profit Organization
- RES - Research Organization
- EDU - Educational Organization
- LEO - Law Enforcement Organization
- PUB - Public Organization
- UKN - Unknown/Other

Member Status:

- ACTIVE - Member is onboarded and active
- INACTIVE - Member is temporarily inactive
- TERMINATED - Member has been offboarded

5.3 User Management

5.3.1 Description

The User Management component provides the Open API for the administration of a basic pool of platform users. The definition of "platform user", in the FAME context, is the following: a *physical person* that is onboarded by an Onboarding Authority or Root Authority (see §4.2.1 for a short description of OA and RA federation member types), and thus receives a Verifiable Onboarding Credential (VOC) that can be stored in a FAME-compatible self-sovereign identity wallet and presented on request as a *proof of onboarding*.

Significant Updates: The User Management component has been enhanced with improved invitation handling, better credential management, and more robust user lifecycle operations. New features include user reinvitation capabilities and improved integration with the credential issuance system.

Full-fledged OAs are *not* expected to use this component: they will typically have their own user management software and will deploy their own integrated solution for the issuance of VOCs. Instead, the User Management component is intended as a simple tool for the FAME *operational governance team* to register and onboard administrators and user support team members.

5.3.2 Technical Specification

5.3.2.1 Public API Interfaces

Onboard User — Starts the onboarding process of a candidate user

API Visibility: PUBLIC - Access Control: authenticated user with admin role

POST `/api/v1.0/users`

Request Body (Content-Type: application/json):

```
{
  "role": "admin|operator|user",
  "country": "string",
  "affiliation": "string",
  "nickname": "string",
  "contact": {
    "firstName": "string",
    "lastName": "string",
    "email": "string",
    "phone": "string"
  }
}
```

Response Body (Content-Type: application/json):

```
{
  "id": "string"
}
```

Response Codes:

- 202: The user record has been created and an invitation message has been sent
- 400: Bad Request: missing or invalid input data
- 401: Unauthorized: missing or invalid authentication
- 403: Forbidden: access denied due to policy
- 500: Internal Server Error: something went wrong, the operation failed

- 502: Bad Gateway: an upstream server was unable to process this request

---

List Users — List users

API Visibility: PUBLIC - Access Control: authenticated user with admin or operator role

GET `/api/v1.0/users`

Query Parameters:

- `iid` - Filter by invitation ID (optional, 22-characters short UUID)
- `rol` - Filter by user role (optional, values: admin, operator, user)
- `aff` - Filter by affiliation PID (optional, 22-characters short UUID)
- `cnt` - Filter by country of residence (optional, ISO 3166-1 alpha-2 code)
- `lname` - Filter by last name (optional, full match only)
- `email` - Filter by email address (optional, full match only)
- `sta` - Filter by user status (optional, values: INVITED, ONBOARDED, OFFBOARDED)

Response Body (Content-Type: application/json):

```
[
  {
    "uid": "string",
    "role": "admin|operator|user",
    "affiliation": "string",
    "lastName": "string",
    "active": boolean
  }
]
```

Response Codes:

- 200: The response body contains an array of references to users matching the given criteria
- 400: Bad Request: missing or invalid input data
- 401: Unauthorized: missing or invalid authentication
- 403: Forbidden: access denied due to policy
- 500: Internal Server Error: something went wrong, the operation failed

---

Retrieve User — Retrieve user

API Visibility: PUBLIC - Access Control: authenticated user with admin or operator role

GET `/api/v1.0/users/{uid}`

Path Parameters:

- `uid` - UID of the user to retrieve

Response Body (Content-Type: application/json):

```
{
  "uid": "string",
  "iid": "string",
  "invited": "string",
  "otp": "string",
  "otpAttempts": number,
  "role": "admin|operator|user",
  "country": "string",
  "affiliation": "string",
  "nickname": "string",
  "firstName": "string",
  "lastName": "string",
  "email": "string",
  "phone": "string",
  "onboarded": "string",
  "offboarded": "string"
}
```

Response Codes:

- 200: The user record has been found and is returned in the response body
- 401: Unauthorized: missing or invalid authentication
- 403: Forbidden: access denied due to policy
- 404: Not Found: the target resource does not exist or is in invalid state
- 500: Internal Server Error: something went wrong, the operation failed

---

Reinvite User — Restarts from scratch the onboarding process of a candidate user

API Visibility: PUBLIC - Access Control: authenticated user with admin role

POST `/api/v1.0/users/{uid}/reinvite`

Path Parameters:

- `uid` - UID of the target user

Response Codes:

- 204: The user record has been updated and a new invitation message has been sent
- 400: Bad Request: missing or invalid input data
- 401: Unauthorized: missing or invalid authentication
- 403: Forbidden: access denied due to policy
- 500: Internal Server Error: something went wrong, the operation failed
- 502: Bad Gateway: an upstream server was unable to process this request

---

Delete User — Deletes a user

API Visibility: PUBLIC - Access Control: authenticated user with admin role

DELETE `/api/v1.0/users/{uid}`

Path Parameters:

- `uid` - UID of the target user

Response Codes:

- 204: The user record has been deleted
- 400: Bad Request: missing or invalid input data

- 401: Unauthorized: missing or invalid authentication
- 403: Forbidden: access denied due to policy
- 404: Not Found: the target resource does not exist or is in invalid state
- 500: Internal Server Error: something went wrong, the operation failed

---

Offboard User — Offboards a user

API Visibility: PUBLIC - Access Control: authenticated user with admin role

POST `/api/v1.0/users/{uid}/offboard`

Path Parameters:

- `uid` - UID of the target user

Response Codes:

- 204: The user record has been updated
- 400: Bad Request: missing or invalid input data
- 401: Unauthorized: missing or invalid authentication
- 403: Forbidden: access denied due to policy
- 404: Not Found: the target resource does not exist or is in invalid state
- 500: Internal Server Error: something went wrong, the operation failed

---

Accept Invitation — Accept onboarding invitation

API Visibility: PUBLIC - Access Control: none (public endpoint)

POST `/api/v1.0/invitations`

Request Body (Content-Type: application/json):

```
{
  "iid": "string",
  "otp": "string"
}
```

Response Body (Content-Type: application/json):

```
{
  "uri": "string",
  "preAuthCode": "string",
  "attemptsLeft": number
}
```

Response Codes:

- 201: The invitation has been accepted and the user is now registered
- 400: Bad Request: missing or invalid input data
- 404: Not Found: the target resource does not exist or is in invalid state
- 500: Internal Server Error: something went wrong, the operation failed
- 502: Bad Gateway: an upstream server was unable to process this request

5.3.2.2 *Data Structures*

User Roles (UROLE domain):
- admin - Platform administrator
- operator - User support (helpdesk) operator
- user - Regular platform user

User Status (USTAT domain):
- INVITED - User has been invited but not yet onboarded
- ONBOARDED - User has completed the onboarding process
- OFFBOARDED - User has been offboarded

5.4 User Support

5.4.1 Description

Note: The User Support component specifications are documented but this functionality is not yet implemented in the current platform release. The component is designed to work on two levels when fully implemented.

Firstly, it will implement a ticketing system that serves as the central hub for all user support activities: platform users can ask for support on FAME-related issues by opening a Support Ticket, and the support team will be able to follow up and track the status of the issue through the same system. This mediation role is extremely important due to the privacy-friendly way in which the FAME Platform is designed, as users are not supposed to disclose their contact information - or even their email address - with anyone outside of their Onboarding Authority.

Secondly, the component implements a User Request queue. Open API operations that are processed asynchronously (e.g., account enrollment operations) use the queue as a messaging channel: the user submits an execution request, the request is dispatched to the Platform service in charge of execution, the service updates the request by adding the execution's outcome, and the user checks the outcome.

5.4.2 Technical Specification

5.4.2.1 *Implemented Interfaces*

Check Request Status — Check Request Status

API Visibility: PUBLIC - Access Control: authenticated user (request owner or admin)

GET `/api/v1.0/requests/{rid}`

Path Parameters:

- `rid` - Request ID

Response Body (Content-Type: application/json):

```
{
  "finalized": "string",
  "status": "string",
  "message": "string"
}
```

Response Codes:

- 200: The found record
- 401: Unauthorized: missing or invalid authentication

- 403: Forbidden: access denied due to policy
- 404: Not Found: the target resource does not exist or is in invalid state
- 500: Internal Server Error: something went wrong, the operation failed

5.4.2.2 Planned Interfaces (Not Yet Implemented)

The following interfaces are documented from the previous specification but are not yet implemented:

- **Open Ticket** — Creates a new Support Ticket
- **Receive Ticket** — Assigns a Support Ticket to an operator
- **Reply to Ticket** — Adds a message to a Support Ticket
- **Suspend Ticket** — Suspends processing of a Support Ticket
- **Close Ticket** — Closes a Support Ticket
- **Reopen Ticket** — Reopens a closed Support Ticket
- **Update Ticket** — Updates Support Ticket information
- **List Tickets** — Retrieves Support Tickets matching criteria
- **Retrieve Ticket** — Retrieves detailed Support Ticket information

For the complete specification of these planned interfaces, please refer to the previous version of this document.

5.4.2.3 Internal Request Queue Interfaces

Enqueue Request — Enqueue Request

API Visibility: INTERNAL

POST `/gov/v1.0/rqueue`

Request Body (Content-Type: application/json):

```
{
  "rid": "string",
  "requestor": "string",
  "payload": "string",
  "message": "string"
}
```

Response Codes:

- 201: Request enqueued successfully

Retrieve Request — Retrieve Request

API Visibility: INTERNAL

GET `/gov/v1.0/rqueue/{rid}`

Path Parameters:

- `rid` - Request ID

Response Body (Content-Type: application/json):

```
{
  "requestor": "string",
  "enqueued": "timestamp",
  "payload": "string",
  "finalized": "timestamp",
  "status": "string",
  "message": "string"
}
```

Finalize Request — Finalize Request

API Visibility: INTERNAL

PUT `/gov/v1.0/rqueue/{rid}`

Path Parameters:

- `rid` - Request ID

Query Parameters:

- `message` - Human-readable status message
- `status` - Request status (PROCESSED|ERROR)

Response Codes:

- 200: Request finalized successfully

5.5 Trading Support

5.5.1 Description

Major Overhaul Notice: The Trading Support component has been entirely overhauled since the previous release. The new implementation provides a streamlined approach to trading account management with enhanced integration capabilities.

The Trading Support component manages the essential infrastructure for trading operations within the FAME marketplace. Trading accounts are fundamental to FAME's data marketplace: they ensure that all supported business models can work through *online transactions*, where access to data assets is unlocked by real-time payments.

A Trading Account is a FAME *extension* to a plain blockchain account. It is based on a regular, Ethereum-style account but has two additional features:

- **Permissioning** - The FAME Blockchain Infrastructure is a *permissioned* network, which means that blockchain accounts must be included in a system-managed *allowlist* to be authorized to execute any smart contract, or to submit blockchain transactions in general.
- **Pseudonymous ownership** - FAME blockchain accounts are not entirely anonymous as in the Ethereum world: they are mapped to the ID of the user who owns them - basically, to a pseudonym that, in case of need, can be traced back to a real identity with the cooperation of the OA who originally onboarded the user.

The component provides capabilities for:

- Granting or revoking permission to operate on individual trading accounts
- Tracking the ownership of trading accounts, linking each to the correct trading party
- Managing FDE token operations (minting, burning, transfers)
- Providing insight into account status and balances

5.5.2 Technical Specification

5.5.2.1 Public API Interfaces

Enroll Account — Starts the enrollment of a new trading account

API Visibility: PUBLIC - Access Control: authenticated user

POST `/api/v1.0/accounts`

Request Body (Content-Type: application/json):

```
{
  "tid": "string"
}
```

Response Codes:

- 202: The request has been accepted and will be processed offline
- 400: Bad Request: missing or invalid input data
- 401: Unauthorized: missing or invalid authentication
- 500: Internal Server Error: something went wrong, the operation failed
- 502: Bad Gateway: an upstream server was unable to process this request

---

Disenroll Account — Permanently disenrolls a trading account

API Visibility: PUBLIC - Access Control: authenticated user (account owner or admin)

DELETE `/api/v1.0/accounts/{tid}`

Path Parameters:

- `tid` - TID of the target trading account

Response Codes:

- 202: The request has been accepted and will be processed offline
- 400: Bad Request: missing or invalid input data
- 401: Unauthorized: missing or invalid authentication
- 404: Not Found: the target resource does not exist or is in invalid state
- 500: Internal Server Error: something went wrong, the operation failed
- 502: Bad Gateway: an upstream server was unable to process this request

---

List Accounts — List trading accounts

API Visibility: PUBLIC - Access Control: authenticated user with admin or operator role

GET `/api/v1.0/accounts`

Query Parameters:

- `usr` - Owner UID (user) to filter accounts by (optional)
- `own` - Owner PID (user affiliation) to filter accounts by (optional)
- `aut` - Onboarding Authority DID (user onboarder) to filter accounts by (optional)
- `sta` - Status to filter accounts by (optional) Values: ACTIVE, INACTIVE, TERMINATED

Response Body (Content-Type: application/json):

```
[
  {
    "tid": "string",
    "owningUser": "string",
    "owningMember": "string",
    "userAuthority": "string",
    "enrolled": "string",
    "disenrolled": "string"
  }
]
```

Response Codes:

- 200: Array of accounts matching the given parameters
- 401: Unauthorized: missing or invalid authentication
- 403: Forbidden: access denied due to policy
- 500: Internal Server Error: something went wrong, the operation failed

---

Retrieve Account — Retrieves a trading account

API Visibility: PUBLIC - Access Control: authenticated user with admin or operator role

GET `/api/v1.0/accounts/{tid}`

Path Parameters:

- `tid` - TID of the target trading account

Response Body (Content-Type: application/json):

```
{
  "tid": "string",
  "owningUser": "string",
  "owningMember": "string",
  "userAuthority": "string",
  "enrolled": "string",
  "disenrolled": "string"
}
```

Response Codes:

- 200: Trading account details
- 400: Bad Request: missing or invalid input data
- 401: Unauthorized: missing or invalid authentication
- 404: Not Found: the target resource does not exist or is in invalid state
- 500: Internal Server Error: something went wrong, the operation failed

---

Credit Account — Mints a given amount of FDE tokens and transfers them to a given trading account

API Visibility: PUBLIC - Access Control: authenticated user with admin or operator role

POST `/api/v1.0/accounts/{tid}/credit`

Path Parameters:

- `tid` - Trading account ID

Request Body (Content-Type: application/json):

```
{
  "amount": "string"
}
```

Response Codes:

- 204: The request has been processed successfully
- 400: Bad Request: missing or invalid input data
- 401: Unauthorized: missing or invalid authentication
- 403: Forbidden: access denied due to policy
- 500: Internal Server Error: something went wrong, the operation failed
- 502: Bad Gateway: an upstream server was unable to process this request

---

Debit Account — Burns a given amount of FDE tokens in a given trading account

API Visibility: PUBLIC - Access Control: authenticated user with admin or operator role

POST `/api/v1.0/accounts/{tid}/debit`

Path Parameters:

- `tid` - Trading account ID

Request Body (Content-Type: application/json):

```
{
  "amount": "string"
}
```

Response Codes:

- 204: The request has been processed successfully
- 400: Bad Request: missing or invalid input data
- 401: Unauthorized: missing or invalid authentication
- 403: Forbidden: access denied due to policy
- 500: Internal Server Error: something went wrong, the operation failed
- 502: Bad Gateway: an upstream server was unable to process this request

5.5.2.2 Internal API Interfaces

Process Account Request — Starts the processing of an account request

API Visibility: INTERNAL

POST `/gov/v1.0/accounts`

Request Body (Content-Type: application/json):

```
{
  "tid": "string",
  "uid": "string",
  "role": "admin|operator|user",
  "did": "string",
  "affiliation": "string",
  "operation": "ENROL|DISENROL|ACTIVATE|DEACTIVATE"
}
```

Response Codes:

- 202: The request has been accepted and will be processed offline
- 400: Bad Request: missing or invalid input data
- 403: Forbidden: access denied due to policy
- 404: Not Found: the target resource does not exist or is in invalid state
- 500: Internal Server Error: something went wrong, the operation failed
- 502: Bad Gateway: an upstream server was unable to process this request

---

Confirm Account Allowed — Confirms account permissioning (callback operation)

API Visibility: INTERNAL

PUT `/gov/v1.0/accounts/{tid}/allowed`

Path Parameters:

- `tid` - TID of the account being enrolled

Response Codes:

- 202: Success
- 400: Bad Request: missing or invalid input data
- 500: Internal Server Error: something went wrong, the operation failed
- 502: Bad Gateway: an upstream server was unable to process this request

---

Confirm Account Disallowed — Confirms account de-permissioning (callback operation)

API Visibility: INTERNAL

PUT `/gov/v1.0/accounts/{tid}/disallowed`

Path Parameters:

- `tid` - TID of the account being disenrolled

Response Codes:

- 202: Success
- 400: Bad Request: missing or invalid input data
- 500: Internal Server Error: something went wrong, the operation failed
- 502: Bad Gateway: an upstream server was unable to process this request

5.5.2.3 Data Structures

Account Status:

- ACTIVE - Account is enrolled and active for trading
- INACTIVE - Account is temporarily inactive
- TERMINATED - Account has been permanently disenrolled

User Roles:

- admin - Platform administrator
- operator - Platform operator
- user - Regular platform user

Account Operations:

- ENROL - Enroll a new trading account
- DISENROL - Disenroll an existing trading account
- ACTIVATE - Activate an inactive account
- DEACTIVATE - Deactivate an active account

5.6 Ecosystem Statistics

5.6.1 Description

New Component: The Ecosystem Statistics component is a new addition that provides basic statistics about the current state of the FAME ecosystem. This component offers a high-level overview of platform activity and growth metrics.

5.6.2 Technical Specification

Get Statistics — Retrieves some basic statistics of the FAME Ecosystem

API Visibility: PUBLIC - Access Control: authenticated user

GET `/api/v1.0/stats`

Response Body (Content-Type: application/json):

```
{
  "members": number,
  "accounts": number,
  "localUsers": number
}
```

Response Codes:

- 200: Returns an object containing the desired ecosystem stats
- 401: Unauthorized: missing or invalid authentication
- 500: Internal Server Error: something went wrong, the operation failed

Response Fields:

- `members` - Number of active federation members
- `accounts` - Number of active trading accounts
- `localUsers` - Number of active users from the Root Onboarding Authority

6 Module Demonstration

This section presents the comprehensive demonstration and validation activities conducted for the enhanced GOV module, showcasing the practical implementation of the architectural improvements and new capabilities introduced in this iteration.

6.1 Full Deployment of the demonstrator

The current iteration of the GOV module represents a substantial advancement over the initial prototype, implementing a comprehensive set of operational governance capabilities that demonstrate the platform's readiness for real-world federation management scenarios. The enhanced demonstrator validates both the technical architecture and the practical applicability of the governance frameworks established throughout the project.

6.1.1 Demonstration Scope and Capabilities

The enhanced MVP demonstrator encompasses the complete spectrum of governance operations required for a federated data marketplace. The demonstration environment successfully validates the following operational scenarios:

Federation Lifecycle Management: The system demonstrates complete member lifecycle operations, from initial onboarding through active management to controlled offboarding. The demonstration includes both traditional administrative pathways and modern external application integration through the FFGA, showcasing the flexibility of the governance framework.

Advanced User Operations: User management demonstrations cover the full spectrum from invitation-based onboarding to sophisticated access control mechanisms. The system validates both internal user management for platform operators and external delegation to federation member authorities, ensuring compliance with privacy and regulatory requirements.

Trading Infrastructure Management: The demonstrator validates the complete trading account ecosystem, including enrollment procedures, permission management, and token operations. Integration with the P&T module demonstrates the separation of concerns between governance and blockchain operations.

Automated Processing Capabilities: The demonstration environment showcases automated workflows for request processing, status tracking, and callback integration, highlighting the system's ability to handle complex multi-module operations seamlessly.

6.1.2 Technical Architecture Validation

The demonstration environment operates on a modern cloud infrastructure that validates the architectural decisions made throughout the development process. The technical validation encompasses:

Microservices Architecture: The GOV module operates as a collection of loosely coupled services, each responsible for specific governance domains. This architecture demonstrates scalability and maintainability while ensuring clear separation of responsibilities.

API-First Design: All functionality is exposed through well-defined REST APIs, enabling both direct integration and external application development. The demonstration validates the effectiveness of the public/internal API separation strategy.

Security Framework Integration: The demonstration environment validates the complete authentication and authorization chain, from VOC-based user identification through blacklist checking to fine-grained permission enforcement.

Cross-Module Integration: The system demonstrates seamless integration with other FAME platform components, validating the internal API patterns and callback mechanisms essential for federated operations.

6.1.3 Performance and Reliability Validation

Extensive testing within the demonstration environment has validated the system's performance characteristics and operational reliability:

Response Time Validation: API operations consistently demonstrate sub-second response times for standard governance operations, with more complex workflows completing within acceptable timeframes.

Concurrent Operation Support: The system successfully handles multiple simultaneous operations across different governance domains without performance degradation or data consistency issues.

Error Handling and Recovery: The demonstration validates comprehensive error handling mechanisms, including graceful degradation scenarios and automatic recovery procedures for transient failures.

Data Integrity Assurance: All operations maintain strict data consistency requirements, with comprehensive audit trail generation supporting compliance and monitoring requirements.

6.2 Advanced Deployment of Server-based Software

The server-based deployment architecture has been substantially enhanced to support the complex integration requirements and operational scenarios of the updated GOV module. The deployment framework provides multiple configuration options to accommodate diverse organizational needs and technical environments.

6.2.1 Containerized Deployment Architecture

The GOV module leverages a sophisticated containerization strategy that ensures consistent operation across diverse deployment environments while maintaining flexibility for specific organizational requirements.

Container Orchestration: The system utilizes modern container orchestration platforms to manage service lifecycle, resource allocation, and network connectivity. This approach ensures reliable operation while providing scalability options for growing federation requirements.

Configuration Management: All deployment-specific parameters are externalized through environment variables and configuration files, enabling rapid deployment across different environments without code modifications.

Service Discovery and Communication: Internal service communication is managed through service mesh patterns, ensuring reliable inter-component communication while maintaining security boundaries.

Health Monitoring and Management: Comprehensive health checking mechanisms ensure early detection of operational issues and enable automated recovery procedures.

6.2.2 Integration Framework Deployment

The enhanced deployment architecture specifically addresses the complex integration requirements of the federated governance environment:

External Application Integration: Deployment configurations support seamless integration with external governance applications like FFGA.

Platform Module Connectivity: Specialized deployment configurations ensure reliable communication with the Provenance & Tracing and the Trading & Monetization modules, including callback endpoint registration and secure API authentication.

PSP Integration Readiness: The deployment framework includes all necessary components for future Payment Service Provider integration, enabling rapid activation when Digital Euro processing becomes available.

Monitoring and Observability: Comprehensive logging, metrics collection, and distributed tracing capabilities provide complete visibility into system operation and performance characteristics.

6.2.3 Security Hardening and Compliance

The deployment architecture incorporates enterprise-grade security measures and compliance capabilities:

Network Security: All communications are secured using industry-standard encryption protocols, with network segmentation ensuring appropriate isolation between different system components.

Access Control Implementation: Multi-layered access control mechanisms ensure that only authorized entities can interact with specific system components, with comprehensive audit logging for all access attempts.

Data Protection Measures: All sensitive data is protected using encryption at rest and in transit, with key management procedures meeting enterprise security requirements.

6.3 Operational Validation and Performance Metrics

Extensive operational validation has been conducted to verify the system's readiness for production deployment and its ability to meet the performance requirements of a federated governance environment.

6.3.1 Functional Validation Results

Comprehensive functional testing has validated all aspects of the GOV module's operational capabilities:

Governance Workflow Validation: All federation member and user lifecycle workflows have been validated under realistic operational conditions, confirming the effectiveness of both traditional and modern governance approaches.

Integration Capability Verification: External application integration capabilities have been thoroughly tested using the FFGA as a primary integration partner, validating the effectiveness of the internal API design.

Cross-Module Operation Validation: All interactions with other FAME platform modules have been tested to ensure reliable operation in the complete federated environment.

Security Framework Verification: The complete security framework, including VOC validation, blacklist checking, and access control enforcement, has been validated under various operational scenarios.

6.3.2 Performance Characteristics

Detailed performance analysis has quantified the system's operational characteristics:

API Performance Metrics: Standard governance operations complete within 100-500 milliseconds, with complex workflows completing within 2-5 seconds depending on external integration requirements.

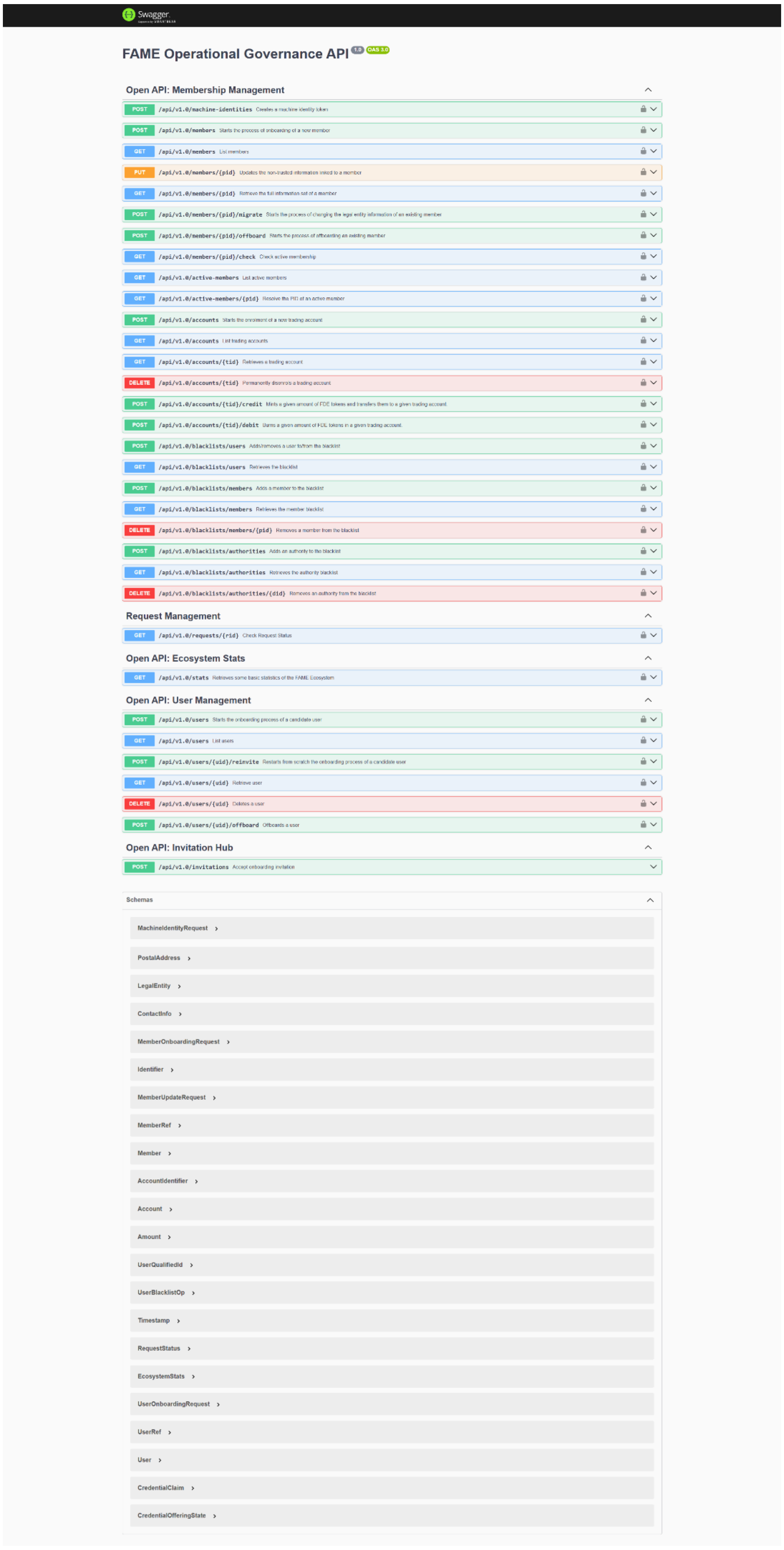
Concurrent User Support: The system reliably supports up to 100 concurrent users for typical governance operations, with graceful performance degradation under higher loads.

Database Performance: Database operations maintain consistent performance characteristics under normal operational loads, with proper indexing ensuring efficient query execution.

Integration Latency: External integrations add minimal latency to operations, with P&T module interactions typically completing within 200-1000 milliseconds depending on blockchain network conditions.

6.4 Online documentation of Open API

Figure 13 shows all the public service endpoints of the GOV API. The documentation of individual endpoints can be accessed by expanding the *accordion* widget on the corresponding row. Some examples are provided in Figures 14 through 22.



Swagger

FAME Operational Governance API 1.0 OAS 3.0

Open API: Membership Management

```
POST    /api/v1.0/machine-identities  Creates a machine identity token
POST    /api/v1.0/members  Starts the process of onboarding of a new member
GET     /api/v1.0/members  List members
PUT     /api/v1.0/members/{pid}  Updates the non-trusted information linked to a member
GET     /api/v1.0/members/{pid}  Retrieve the full information set of a member
POST    /api/v1.0/members/{pid}/migrate  Starts the process of changing the legal entity information of an existing member
POST    /api/v1.0/members/{pid}/offboard  Starts the process of offboarding an existing member
GET     /api/v1.0/members/{pid}/check  Check active membership
GET     /api/v1.0/active-members  List active members
GET     /api/v1.0/active-members/{pid}  Resolve the PID of an active member
POST    /api/v1.0/accounts  Starts the enrolment of a new trading account
GET     /api/v1.0/accounts  List trading accounts
GET     /api/v1.0/accounts/{tid}  Retrieves a trading account
DELETE  /api/v1.0/accounts/{tid}  Permanently disenrols a trading account
POST    /api/v1.0/accounts/{tid}/credit  Mints a given amount of FDE tokens and transfers them to a given trading account.
POST    /api/v1.0/accounts/{tid}/debit  Burns a given amount of FDE tokens in a given trading account.
POST    /api/v1.0/blacklists/users  Adds/removes a user to/from the blacklist
GET     /api/v1.0/blacklists/users  Retrieves the blacklist
POST    /api/v1.0/blacklists/members  Adds a member to the blacklist
GET     /api/v1.0/blacklists/members  Retrieves the member blacklist
DELETE  /api/v1.0/blacklists/members/{pid}  Removes a member from the blacklist
POST    /api/v1.0/blacklists/authorities  Adds an authority to the blacklist
GET     /api/v1.0/blacklists/authorities  Retrieves the authority blacklist
DELETE  /api/v1.0/blacklists/authorities/{did}  Removes an authority from the blacklist
```

Request Management

```
GET     /api/v1.0/requests/{rid}  Check Request Status
```

Open API: Ecosystem Stats

```
GET     /api/v1.0/stats  Retrieves some basic statistics of the FAME Ecosystem
```

Open API: User Management

```
POST    /api/v1.0/users  Starts the onboarding process of a candidate user
GET     /api/v1.0/users  List users
POST    /api/v1.0/users/{uid}/reinvite  Restarts from scratch the onboarding process of a candidate user
GET     /api/v1.0/users/{uid}  Retrieve user
DELETE  /api/v1.0/users/{uid}  Deletes a user
POST    /api/v1.0/users/{uid}/offboard  Offboards a user
```

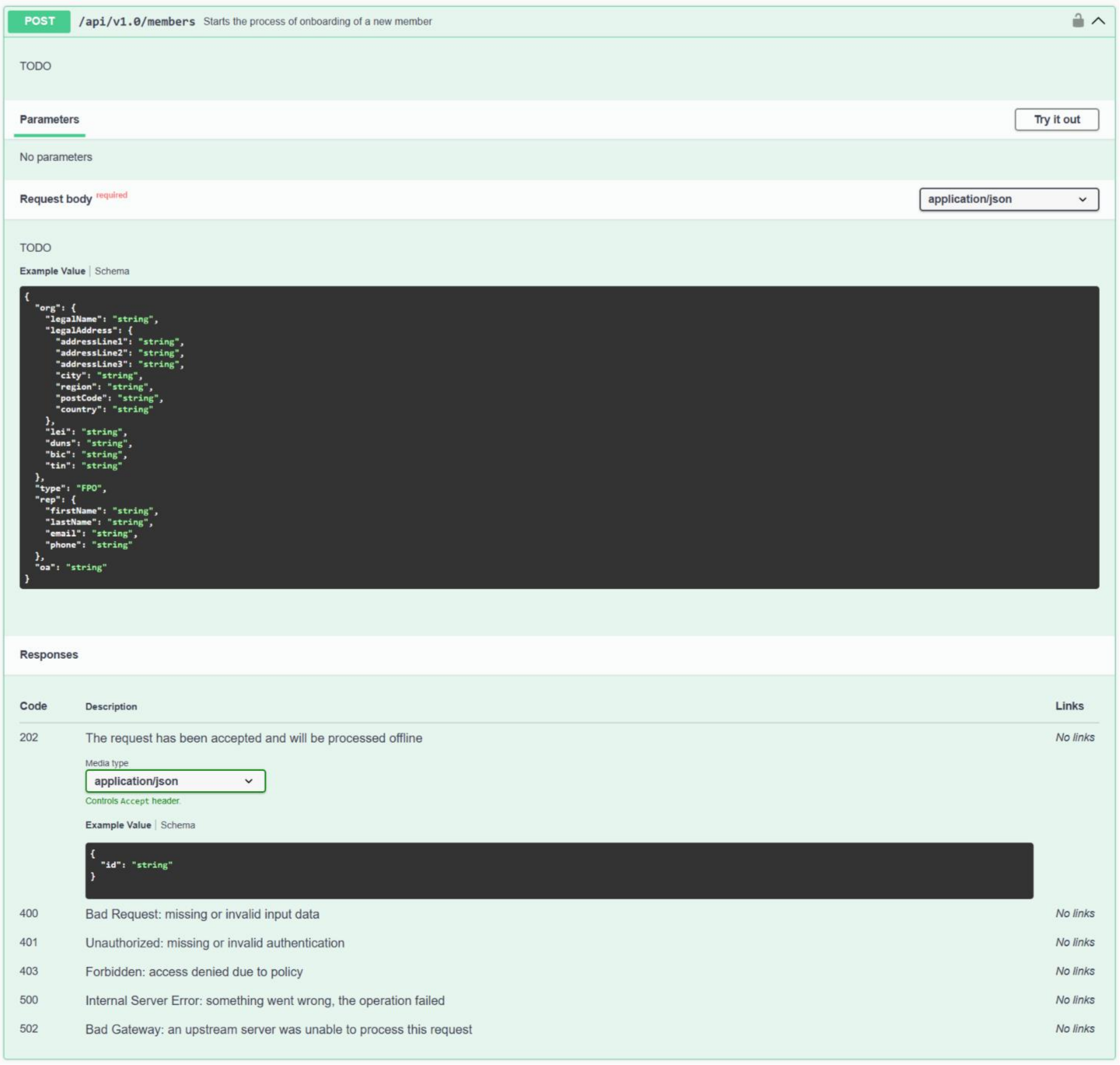
Open API: Invitation Hub

```
POST    /api/v1.0/invitations  Accept onboarding invitation
```

Schemas

- MachineIdentityRequest
- PostalAddress
- LegalEntity
- ContactInfo
- MemberOnboardingRequest
- Identifier
- MemberUpdateRequest
- MemberRef
- Member
- AccountIdentifier
- Account
- Amount
- UserQualifiedId
- UserBlacklistOp
- Timestamp
- RequestStatus
- EcosystemStats
- UserOnboardingRequest
- UserRef
- User
- CredentialClaim
- CredentialOfferingState

Figure 13 - Operational Governance Open API overview (online Swagger documentation)



POST `/api/v1.0/members` Starts the process of onboarding of a new member

TODO

Parameters

Try it out

No parameters

Request body required

application/json

TODO

Example Value | Schema

```
{
  "org": {
    "legalName": "string",
    "legalAddress": {
      "addressLine1": "string",
      "addressLine2": "string",
      "addressLine3": "string",
      "city": "string",
      "region": "string",
      "postCode": "string",
      "country": "string"
    },
    "lei": "string",
    "duns": "string",
    "bic": "string",
    "tin": "string"
  },
  "type": "FPO",
  "rep": {
    "firstName": "string",
    "lastName": "string",
    "email": "string",
    "phone": "string"
  },
  "oa": "string"
}
```

Responses

Code	Description	Links
202	The request has been accepted and will be processed offline	*No links*
400	Bad Request: missing or invalid input data	*No links*
401	Unauthorized: missing or invalid authentication	*No links*
403	Forbidden: access denied due to policy	*No links*
500	Internal Server Error: something went wrong, the operation failed	*No links*
502	Bad Gateway: an upstream server was unable to process this request	*No links*

Media type

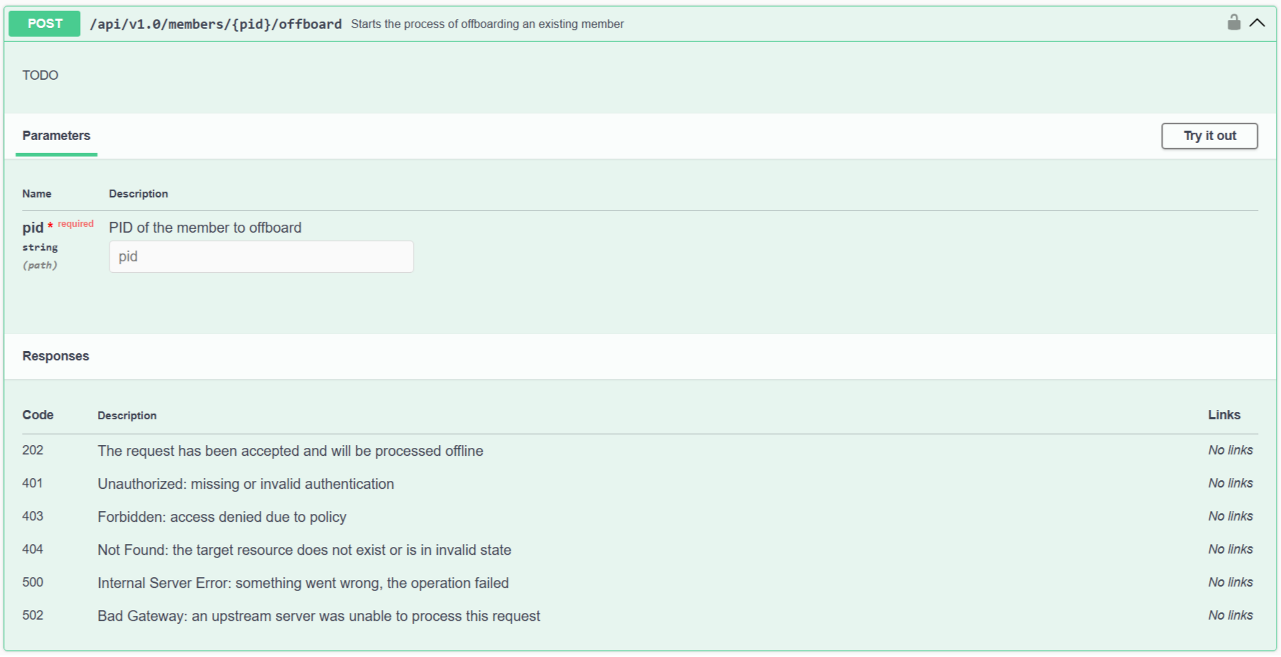
application/json

Controls Accept header.

Example Value | Schema

```
{
  "id": "string"
}
```

Figure 14 - Onboard Member service endpoint



POST `/api/v1.0/members/{pid}/offboard` Starts the process of offboarding an existing member

TODO

Parameters

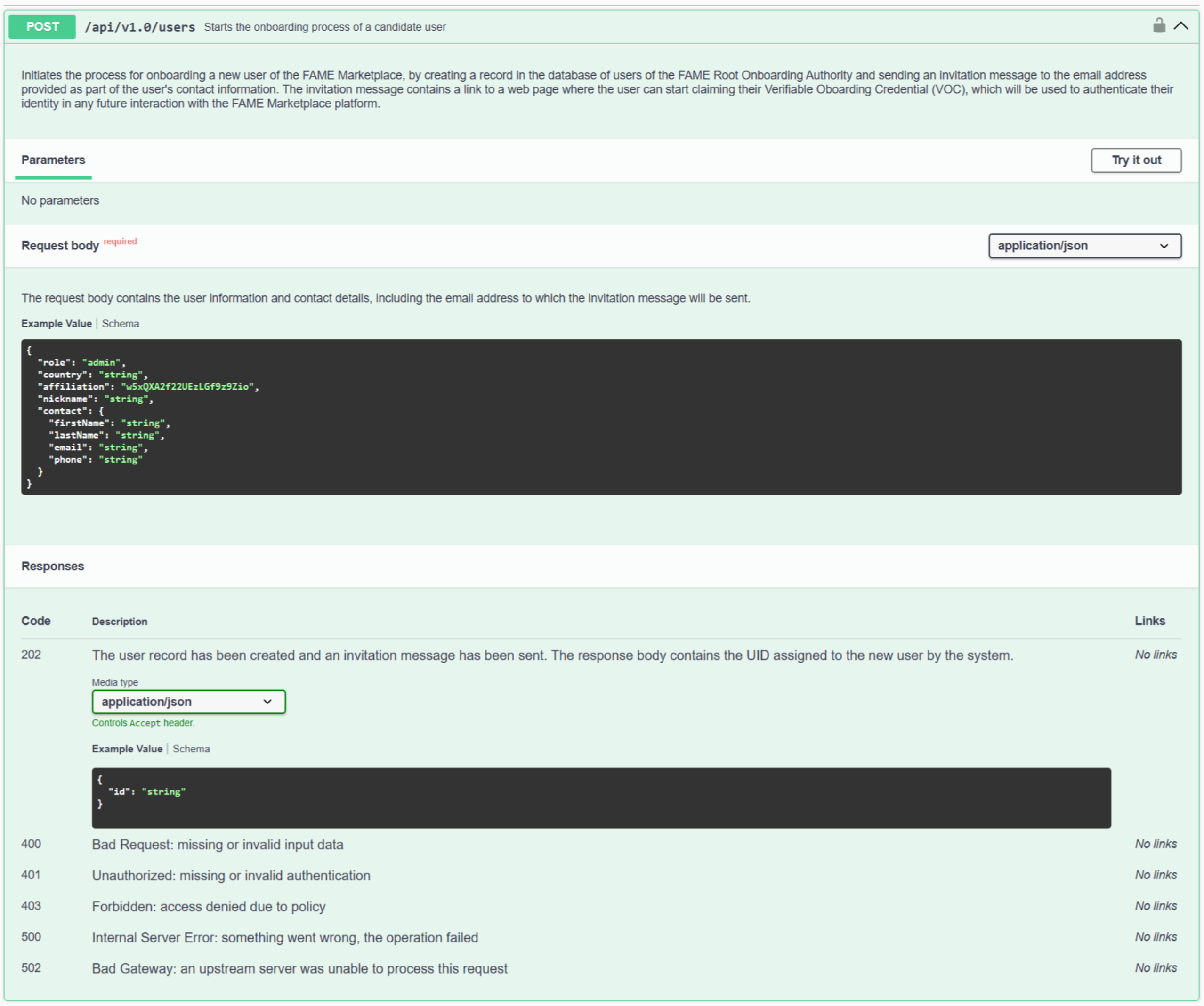
Try it out

Name	Description
pid * required string *(path)*	PID of the member to offboard pid

Responses

Code	Description	Links
202	The request has been accepted and will be processed offline	*No links*
401	Unauthorized: missing or invalid authentication	*No links*
403	Forbidden: access denied due to policy	*No links*
404	Not Found: the target resource does not exist or is in invalid state	*No links*
500	Internal Server Error: something went wrong, the operation failed	*No links*
502	Bad Gateway: an upstream server was unable to process this request	*No links*

Figure 15 - Offboard Member service endpoint



POST `/api/v1.0/users` Starts the onboarding process of a candidate user

Initiates the process for onboarding a new user of the FAME Marketplace, by creating a record in the database of users of the FAME Root Onboarding Authority and sending an invitation message to the email address provided as part of the user's contact information. The invitation message contains a link to a web page where the user can start claiming their Verifiable Oboarding Credential (VOC), which will be used to authenticate their identity in any future interaction with the FAME Marketplace platform.

Parameters

Try it out

No parameters

Request body required

application/json

The request body contains the user information and contact details, including the email address to which the invitation message will be sent.

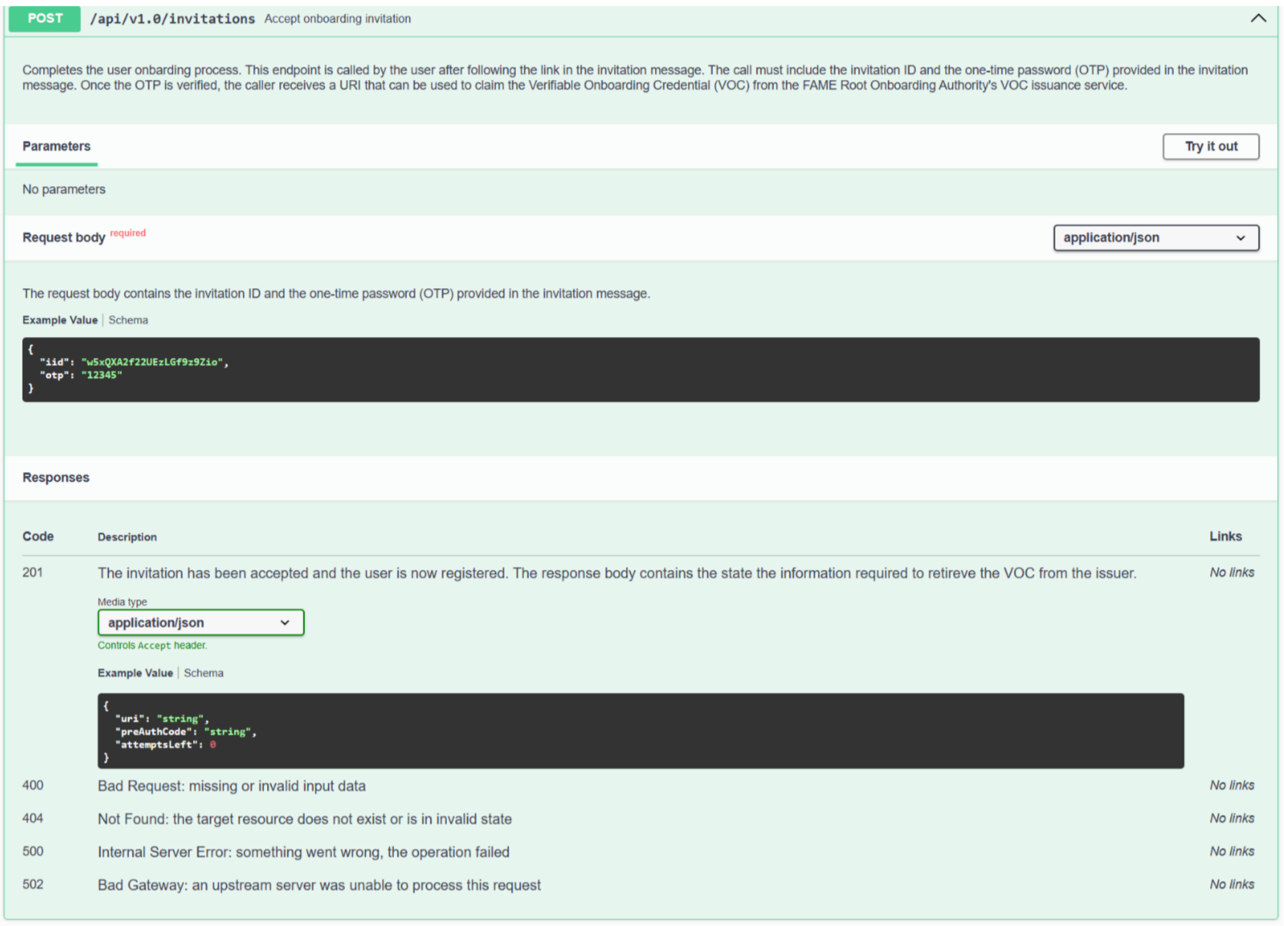
Example Value | Schema

```
{
  "role": "admin",
  "country": "string",
  "affiliation": "w5xQXA2f22UEzLGf9z9Zio",
  "nickname": "string",
  "contact": {
    "firstName": "string",
    "lastName": "string",
    "email": "string",
    "phone": "string"
  }
}
```

Responses

Code	Description	Links
202	The user record has been created and an invitation message has been sent. The response body contains the UID assigned to the new user by the system. Media type application/json Controls Accept header. Example Value | Schema `{ "id": "string" }`	*No links*
400	Bad Request: missing or invalid input data	*No links*
401	Unauthorized: missing or invalid authentication	*No links*
403	Forbidden: access denied due to policy	*No links*
500	Internal Server Error: something went wrong, the operation failed	*No links*
502	Bad Gateway: an upstream server was unable to process this request	*No links*

Figure 16 - Invite User service endpoint



POST `/api/v1.0/invitations` Accept onboarding invitation

Completes the user onbarding process. This endpoint is called by the user after following the link in the invitation message. The call must include the invitation ID and the one-time password (OTP) provided in the invitation message. Once the OTP is verified, the caller receives a URI that can be used to claim the Verifiable Onboarding Credential (VOC) from the FAME Root Onboarding Authority's VOC issuance service.

Parameters Try it out

No parameters

Request body required application/json

The request body contains the invitation ID and the one-time password (OTP) provided in the invitation message.

Example Value | Schema

```
{
  "iid": "w5xQXA2f22UEzLGf9z9Zio",
  "otp": "12345"
}
```

Responses

Code	Description	Links
201	The invitation has been accepted and the user is now registered. The response body contains the state the information required to retireve the VOC from the issuer.	No links
400	Bad Request: missing or invalid input data	No links
404	Not Found: the target resource does not exist or is in invalid state	No links
500	Internal Server Error: something went wrong, the operation failed	No links
502	Bad Gateway: an upstream server was unable to process this request	No links

Media type

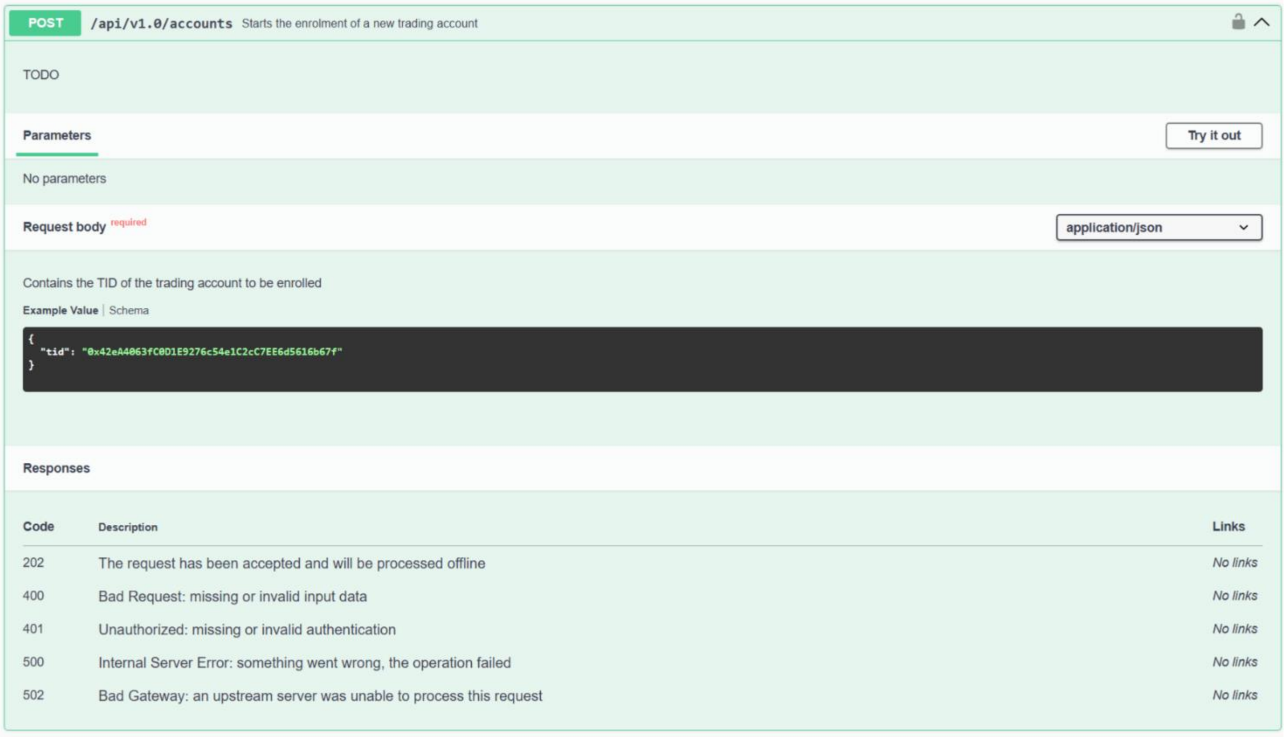
application/json

Controls `Accept` header.

Example Value | Schema

```
{
  "uri": "string",
  "preAuthCode": "string",
  "attemptsLeft": 0
}
```

Figure 17 - Claim Onboarding Credential service endpoint



POST `/api/v1.0/accounts` Starts the enrolment of a new trading account

TODO

Parameters Try it out

No parameters

Request body required application/json

Contains the TID of the trading account to be enrolled

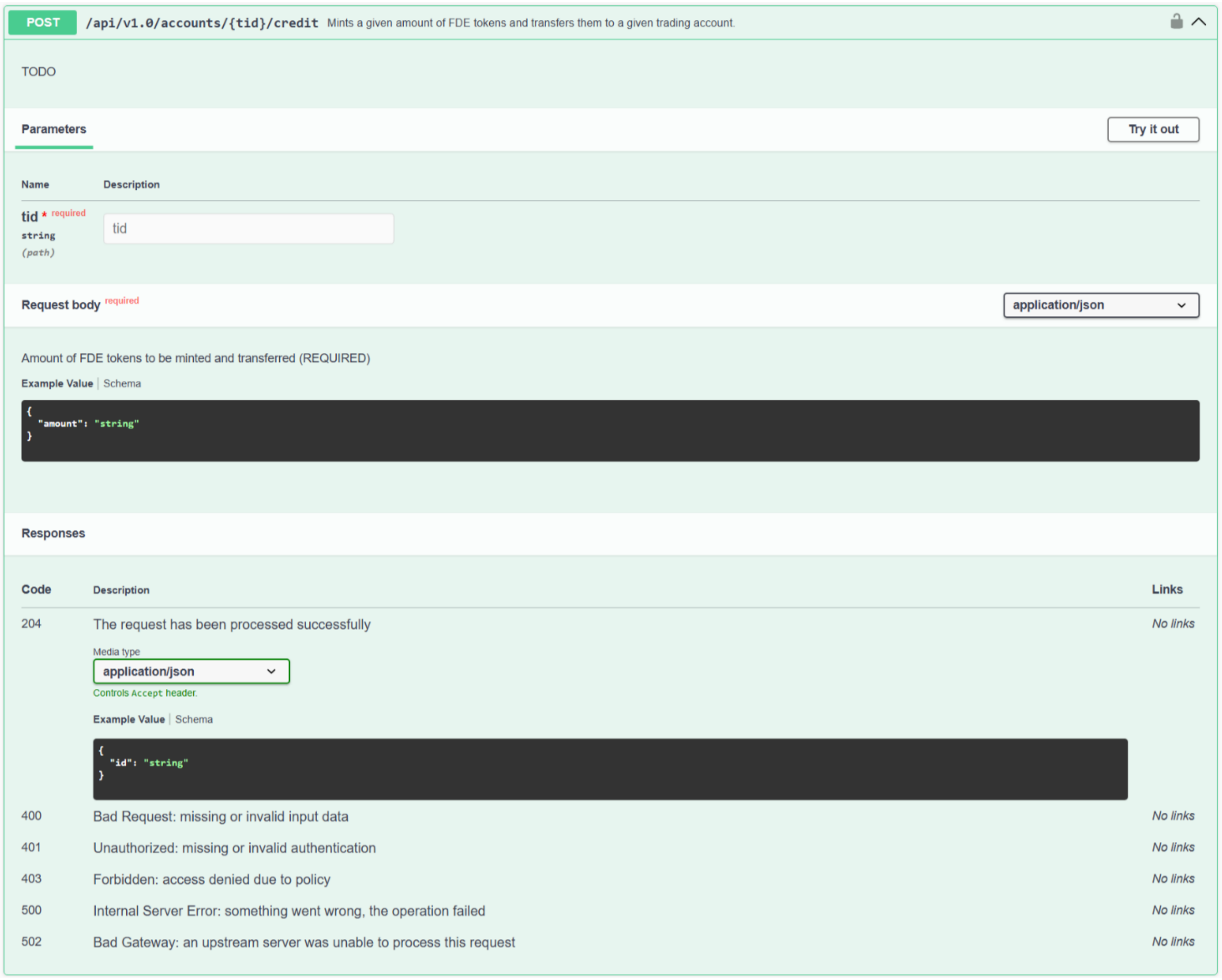
Example Value | Schema

```
{
  "tid": "0x42eA4063fC0D1E9276c54e1C2cC7EE6d5616b67f"
}
```

Responses

Code	Description	Links
202	The request has been accepted and will be processed offline	No links
400	Bad Request: missing or invalid input data	No links
401	Unauthorized: missing or invalid authentication	No links
500	Internal Server Error: something went wrong, the operation failed	No links
502	Bad Gateway: an upstream server was unable to process this request	No links

Figure 18 - Enroll Trading Account service endpoint



POST /api/v1.0/accounts/{tid}/credit Mints a given amount of FDE tokens and transfers them to a given trading account.

TODO

Parameters

Name	Description
tid * required string (path)	tid

Request body required — application/json

Amount of FDE tokens to be minted and transferred (REQUIRED)

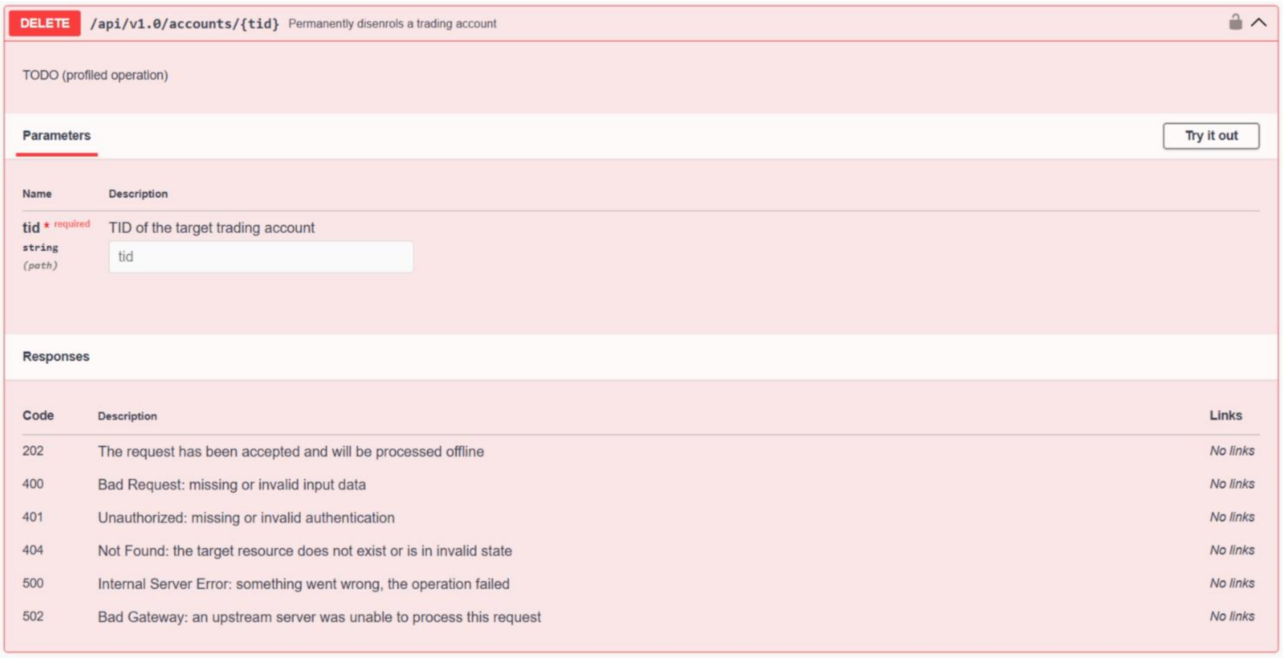
Example Value | Schema

```
{
  "amount": "string"
}
```

Responses

Code	Description	Links
204	The request has been processed successfully Media type: application/json Controls Accept header. Example Value | Schema `{ "id": "string" }`	No links
400	Bad Request: missing or invalid input data	No links
401	Unauthorized: missing or invalid authentication	No links
403	Forbidden: access denied due to policy	No links
500	Internal Server Error: something went wrong, the operation failed	No links
502	Bad Gateway: an upstream server was unable to process this request	No links

Figure 19 - Mint FDE Tokens service endpoint



DELETE /api/v1.0/accounts/{tid} Permanently disenrols a trading account

TODO (profiled operation)

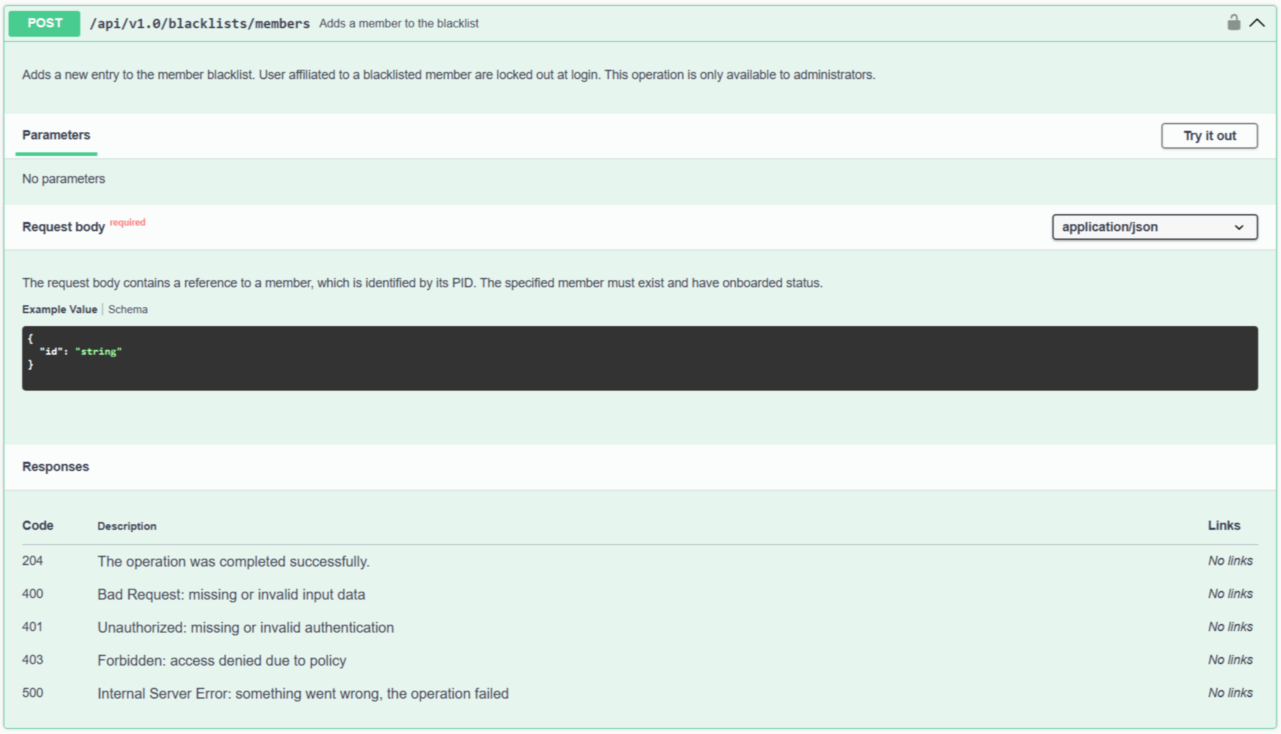
Parameters

Name	Description
tid * required string (path)	TID of the target trading account tid

Responses

Code	Description	Links
202	The request has been accepted and will be processed offline	No links
400	Bad Request: missing or invalid input data	No links
401	Unauthorized: missing or invalid authentication	No links
404	Not Found: the target resource does not exist or is in invalid state	No links
500	Internal Server Error: something went wrong, the operation failed	No links
502	Bad Gateway: an upstream server was unable to process this request	No links

Figure 20 - Disenroll Trading Account service endpoint



POST /api/v1.0/blacklists/members Adds a member to the blacklist

Adds a new entry to the member blacklist. User affiliated to a blacklisted member are locked out at login. This operation is only available to administrators.

Parameters

No parameters

Request body required — application/json

The request body contains a reference to a member, which is identified by its PID. The specified member must exist and have onboarded status.

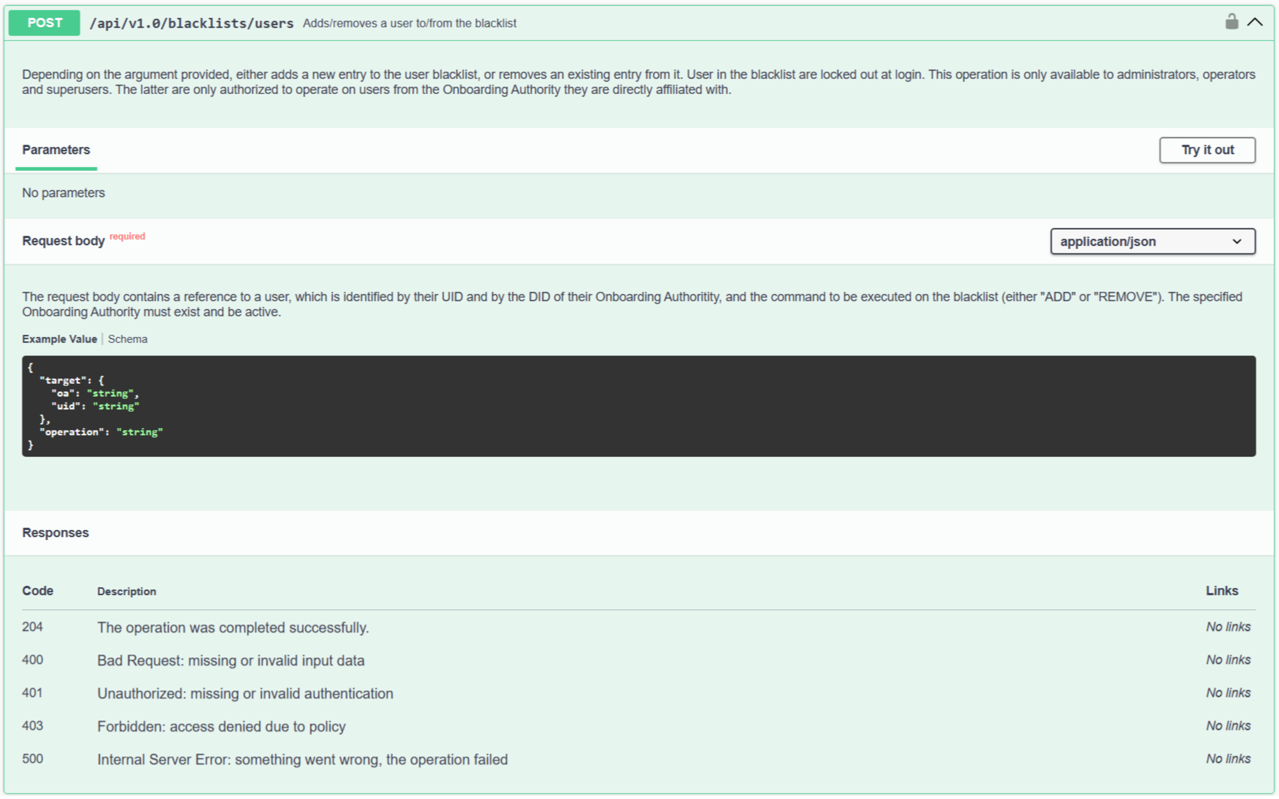
Example Value | Schema

```
{
  "id": "string"
}
```

Responses

Code	Description	Links
204	The operation was completed successfully.	No links
400	Bad Request: missing or invalid input data	No links
401	Unauthorized: missing or invalid authentication	No links
403	Forbidden: access denied due to policy	No links
500	Internal Server Error: something went wrong, the operation failed	No links

Figure 21 - Blacklist Member service endpoint



POST /api/v1.0/blacklists/users Adds/removes a user to/from the blacklist

Depending on the argument provided, either adds a new entry to the user blacklist, or removes an existing entry from it. User in the blacklist are locked out at login. This operation is only available to administrators, operators and superusers. The latter are only authorized to operate on users from the Onboarding Authority they are directly affiliated with.

Parameters

No parameters

Request body required — application/json

The request body contains a reference to a user, which is identified by their UID and by the DID of their Onboarding Authoritity, and the command to be executed on the blacklist (either "ADD" or "REMOVE"). The specified Onboarding Authority must exist and be active.

Example Value | Schema

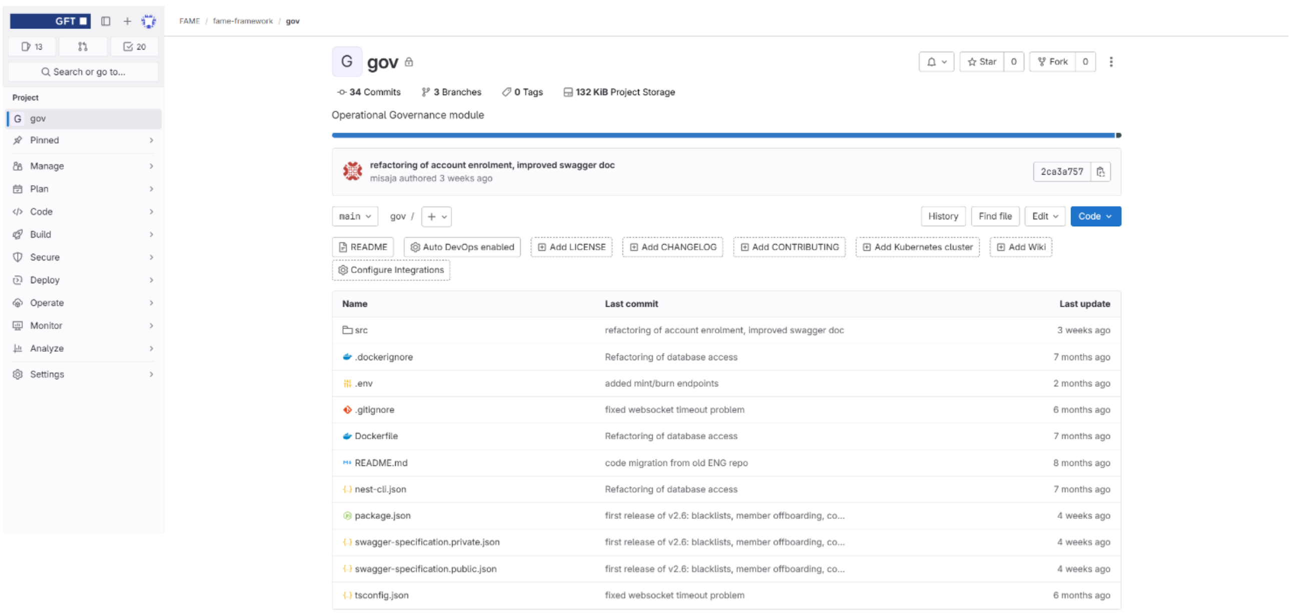
```
{
  "target": {
    "oa": "string",
    "uid": "string"
  },
  "operation": "string"
}
```

Responses

Code	Description	Links
204	The operation was completed successfully.	No links
400	Bad Request: missing or invalid input data	No links
401	Unauthorized: missing or invalid authentication	No links
403	Forbidden: access denied due to policy	No links
500	Internal Server Error: something went wrong, the operation failed	No links

Figure 22 - Blacklist User service endpoint

The GOV module is a modern, modular NestJS/TypeScript application that was developed following industry-standard best practices. Its source code is maintained on a private GitLab repository (see Figure 23) that is managed by the GFT project partner. By the end of the project, the source code will be made publicly available under the Apache 2.0 Open-Source license.



G gov

34 Commits · 3 Branches · 0 Tags · 132 KiB Project Storage

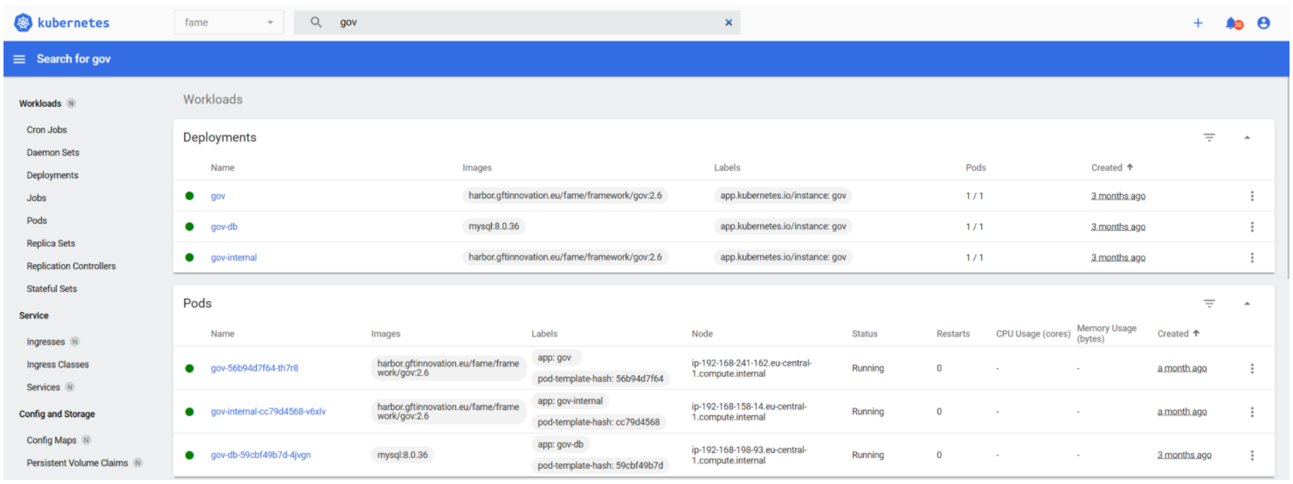
Operational Governance module

refactoring of account enrolment, improved swagger doc
misaja authored 3 weeks ago
2ca3a757

Name	Last commit	Last update
src	refactoring of account enrolment, improved swagger doc	3 weeks ago
.dockerignore	Refactoring of database access	7 months ago
.env	added mint/burn endpoints	2 months ago
.gitignore	fixed websocket timeout problem	6 months ago
Dockerfile	Refactoring of database access	7 months ago
README.md	code migration from old ENG repo	8 months ago
nest-cli.json	Refactoring of database access	7 months ago
package.json	first release of v2.6: blacklists, member offboarding, co...	4 weeks ago
swagger-specification.private.json	first release of v2.6: blacklists, member offboarding, co...	4 weeks ago
swagger-specification.public.json	first release of v2.6: blacklists, member offboarding, co...	4 weeks ago
tsconfig.json	fixed websocket timeout problem	6 months ago

Figure 23 - GitLab source code repository for GOV module

The GOV runtime is distributed as a collection of Docker containers. The container images are built in such a way that all the information required for integration with other modules (e.g., the network address of service endpoints exposed by Provenance & Tracing and by Trading & Monetization) is provided at deployment time as environment variables. This mitigates cross-module dependencies and allows for the module to be easily relocated anywhere in case of need. As proof of this claim, at the end of 2024 the entire FAME Platform, including the GOV module, was migrated from its old hosting environment (a Cloud server operated by ENG) to a more scalable and robust facility based on AWS/Kubernetes. Figure 24 is a screenshot of the Kubernetes Dashboard while focused on the GOV components.



Workloads

Deployments

Name	Images	Labels	Pods	Created
gov	harbor.gftinnovation.eu/fame/framework/gov:2.6	app.kubernetes.io/instance: gov	1 / 1	3 months ago
gov-db	mysql:8.0.36	app.kubernetes.io/instance: gov	1 / 1	3 months ago
gov-internal	harbor.gftinnovation.eu/fame/framework/gov:2.6	app.kubernetes.io/instance: gov	1 / 1	3 months ago

Pods

Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created
gov-56b94d7f64-th7r8	harbor.gftinnovation.eu/fame/framework/gov:2.6	app: gov pod-template-hash: 56b94d7f64	ip-192-168-241-162.eu-central-1.compute.internal	Running	0	-	-	a month ago
gov-internal-cc79d4568-v6xlv	harbor.gftinnovation.eu/fame/framework/gov:2.6	app: gov-internal pod-template-hash: cc79d4568	ip-192-168-158-14.eu-central-1.compute.internal	Running	0	-	-	a month ago
gov-db-59cbf49b7d-4jvgn	mysql:8.0.36	app: gov-db pod-template-hash: 59cbf49b7d	ip-192-168-198-93.eu-central-1.compute.internal	Running	0	-	-	3 months ago

Figure 24 - Kubernetes Dashboard for managing GOV runtime components

7 Conclusions

This deliverable presents an updated view of the operational and business models for the FAME Marketplace, building on the foundations established in the previous version. It reflects progress made under Task T4.5, focusing on the refinement of the hybrid business model strategy and the evolution of the GOV module based on feedback from pilot deployments and ongoing development activities.

The GOV module continues to serve as the core management and governance entry point of the FAME platform. It includes five main components: Federation Management, User Management, User Support, Trading Support and Ecosystem Statistics. In this iteration, each of these components has been further specified and technically enhanced to improve integration, usability, and operational robustness.

The selected business models—subscription-based, pay-as-you-go (PAYG), pay-as-you-use (PAYU), Data-as-a-Service (DaaS), and freemium—have been validated through continued pilot engagement. These models are designed to ensure the marketplace remains adaptable, financially viable, and aligned with the varying needs of different stakeholders.

This deliverable represents a significant advancement in the maturity of the GOV module from a technical perspective. In order to facilitate smooth integration, modular expansion, and scalable operation, its components have been thoroughly described with enhanced APIs and technical documentation. Important processes have been streamlined and brought into compliance with platform-wide governance and security standards, such as account enrollment, the onboarding and offboarding of federation members and users, and the purchase of FDE coins.

These developments were validated through deployments in both server-based and blockchain-compatible environments. Leveraging containerized architectures, integration frameworks, and security hardening practices, the GOV module demonstrated its readiness for scalable, replicable, and compliant adoption. Performance validation confirmed its operational viability under real-world conditions.

In parallel, the integration of Dataspaces and their associated business model building blocks has established a robust framework for secure, interoperable, and monetized data sharing within the federated marketplace. This foundation ensures that data governance, policy enforcement, and flexible monetization coexist seamlessly, supporting trust and sustainability across diverse data ecosystems.

Looking ahead, future work will focus on the progressive refinement and implementation of the selected business models, further expansion of the GOV module's functionalities, and the integration of advanced usage metrics to support dynamic pricing and governance mechanisms. Continued engagement with pilot activities and stakeholder feedback will guide an iterative development approach, ensuring that the platform evolves responsively in line with emerging technological trends, user needs, and regulatory requirements.

References

1. FAME Deliverable D4.3 Operational Models, Business Models and Governance I https://zenodo.org/records/15310303
2. FAME Business Framework Documentation https://zenodo.org/records/15591777
3. FAME Legal Framework Documentation https://zenodo.org/records/15591094
4. FAME Consortium, (2024), Token-Based Monetization in Federated Ecosystems, FAME Consortium White Paper Available at: https://www.fame-horizon.eu/scientific-publications-and-whitepapers/
5. FAME Deliverable D4.1 – Blockchain-based Data Provenance Infrastructure (2025) https://zenodo.org/records/16601858
6. FAME Deliverable D2.6-Technical Specifications and Platform Architecture II (Section 8.2.3 FAME SA Mapping to DSSC Building Blocks , 2024) https://zenodo.org/records/15341321
7. Sadhukhan, P., & Gupta, S. (2025). A graph theoretic approach to assess quality of data for classification task. Data & Knowledge Engineering, 158, 102421.
8. Gupta, S., Peliova, J., Sadhukhan, P., Kumar, P., Vasilakopoulou, P., Pappas, I. Conceptualizing Similarity Measurement in Data Marketplaces. UK Academy for Information Systems (UKAIS) Annual Conference 2025, Newcastle University Business School , U.K. (2025)
9. Data Spaces Support Centre (DSSC), (2024), Blueprint and Building Blocks Handbook, DSSC, [Online] Available at: Data Spaces Blueprint v1.0 - Blueprint v1.0 - Data Spaces Support Centre (Accessed: July 29, 2025).
10. International Data Spaces Association (IDSA), (2022), IDS Reference Architecture Model Version 4.0, IDSA, Berlin. [Online] Available at: https://internationaldataspaces.org/publications/most-important-documents/ (Accessed: July 29, 2025).
11. GAIA-X Association, (2022), GAIA-X Trust Framework and Architecture Document, GAIA-X, Brussels. [Online] Available at: https://docs.gaia-x.eu/technical-committee/architecture-document/latest/ (Accessed: July 29, 2025).
12. European Commission, (2022), Data Governance Act (Regulation (EU) 2022/868), Publications Office of the European Union, Brussels. [Online] Available at: https://digital-strategy.ec.europa.eu/en/policies/data-governance-act (Accessed: July 29, 2025).
13. European Commission, (2023), Data Act (Regulation (EU) 2023/2854), Publications Office of the European Union, Brussels. [Online] Available at: https://digital-strategy.ec.europa.eu/en/policies/data-act (Accessed: July 29, 2025).
14. González, M. and Spichiger, L., (2025), Privacy-Preserving Self-Sovereign Identity under eIDAS 2.0: Challenges and Opportunities, arXiv preprint. [Online] Available at: https://arxiv.org/abs/2502.02520 (Accessed: July 29, 2025).

Attachments

A) Competitor Landscape Appendix

Competitor	Operating Model	Sector Focus	Key Characteristics	Reference
Amazon AWS Data Exchange	Centralized	Cross-sector (B2B, Enterprise)	Large-scale cloud-native data sharing tightly integrated with AWS cloud infrastructure; trusted brand for enterprise IT; centralized governance.	aws.amazon.com/data-exchange
Snowflake Marketplace	Centralized SaaS	Enterprise SaaS	Data sharing as part of its cloud data warehouse; seamless integration with analytics workflows; strong in financial services and large enterprise customers.	snowflake.com/data-marketplace
Ocean Protocol	Decentralized (Web3)	Cross-domain	Blockchain-based, decentralized data marketplace protocol; smart contracts, tokenization (OCEAN token); supports federated data economy vision.	oceanprotocol.com
Datum	Decentralized Personal Data	Personal Data Economy	Focus on empowering individuals to control, monetize and share their personal data; consumer-facing monetization models.	datum.org
Amazon AWS Data Exchange	Centralized	Cross-sector (B2B, Enterprise)	Large-scale cloud-native data sharing tightly	aws.amazon.com/data-exchange

			integrated with AWS cloud infrastructure; trusted brand for enterprise IT; centralized governance.	
Snowflake Marketplace	Centralized SaaS	Enterprise SaaS	Data sharing as part of its cloud data warehouse; seamless integration with analytics workflows; strong in financial services and large enterprise customers.	snowflake.com/data-marketplace
Ocean Protocol	Decentralized (Web3)	Cross-domain	Blockchain-based, decentralized data marketplace protocol; smart contracts, tokenization (OCEAN token); supports federated data economy vision.	oceanprotocol.com
Datum	Decentralized Personal Data	Personal Data Economy	Focus on empowering individuals to control, monetize and share their personal data; consumer-facing monetization models.	datum.org

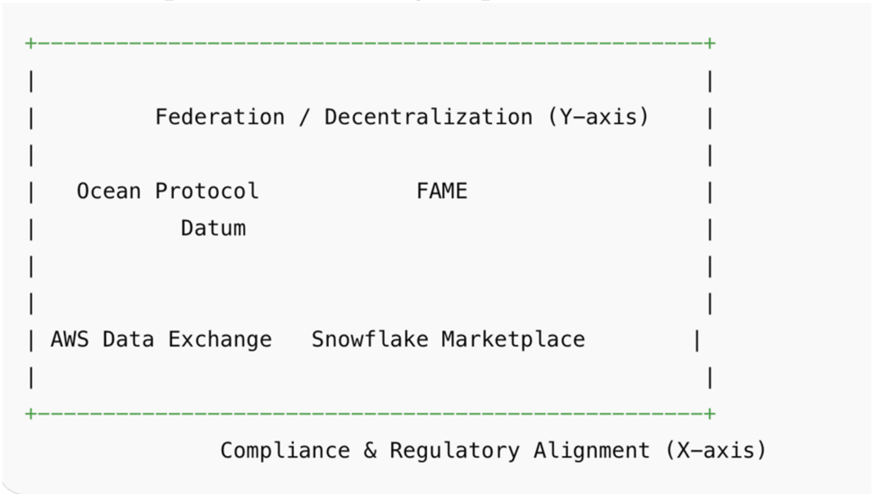
Table 7 Attachments: Overview of Competitors in the Data Marketplace Ecosystem

B) Competitor Classification Map

Model	Key Players
Centralized Platforms	AWS Data Exchange, Snowflake
Decentralized Protocols	Ocean Protocol, Datum
Federated & Compliance-Centric	FAME

Table 8 Attachments: Competitor Classification Map

C) Competitive Positioning Map



```
+---------------------------------------------------+
|                                                    |
|         Federation / Decentralization (Y-axis)    |
|                                                    |
|   Ocean Protocol            FAME                   |
|           Datum                                    |
|                                                    |
|                                                    |
| AWS Data Exchange   Snowflake Marketplace        |
|                                                    |
+---------------------------------------------------+
               Compliance & Regulatory Alignment (X-axis)
```

- Bottom-left quadrant: centralized but low compliance focus.
- Top-right quadrant: FAME’s unique competitive positioning as highly federated, highly compliant.