TOKENIZED ECONOMIES

In collaboration with

Caisse des Dépôts GROUPE

November 2024

Unlocking Scalability and Interoperability in Tokenization Systems

Working Paper N°1 | Chapter 1

Open-source, Composable, and Secure Smart Contract Architectures

FEVERTOKENS

tokenizedeconomies.org

Foreword

The emergence of tokenization represents a monumental shift in how financial assets are managed, traded, and secured. The integration of open-source, composable, and secure architectures is a critical foundation for this transformation. This report, the first in a series titled "Unlocking Scalability and Interoperability in Tokenization Systems," serves as a pivotal step towards rethinking and restructuring tokenization systems beyond the confines of single-stakeholder paradigms.

FeverTokens and Caisse des Dépôts (CDC) are united by a broad vision to integrate all ecosystem players and reduce the barriers to entry for developing and running tokenization systems at scale. Our collaboration under the Tokenized Economies Institute aims to foster an environment where scalability and interoperability are not mere aspirations but achievable standards. The cost and technical barriers that have historically impeded the development of next-generation infrastructure for capital markets must be lowered. This will allow innovators to focus on what truly matters: the standardization of data and interconnectivity, the sophistication of financial engineering, and the governance structures necessary to realize the full potential of tokenization.

This first chapter will introduce the core principles and attributes of scalable tokenization systems. Subsequent chapters will then recapitulate and showcase several industry initiatives that have successfully embraced open, ecosystemic, and interoperable approaches to tokenization. As we navigate the complexities of integrating diverse technological and regulatory frameworks, it is imperative to adhere to a set of core principles. These principles must ensure that tokenization systems are not only technically sound but also aligned with business needs and operational, legal, and compliance constraints. The goal is to create a cohesive ecosystem where every component, from smart contracts to governance protocols, is designed to support the broader ambitions of tokenization.

The collaborative effort between FeverTokens and CDC embodies a commitment to a future where tokenized economies are accessible, efficient, and secure. We invite all stakeholders—developers, financial institutions, regulators, and investors—to join us on this journey. Together, we can unlock the true potential of tokenization, driving innovation and creating a more inclusive and dynamic financial ecosystem.

Nadia Filali

Zakaryae Boudi

1. Foundational Attributes of Tokenization Systems

Amidst the swift advancements in digital finance, tokenization emerges as a transformative force redefining financial markets and asset life-cycles. Essential attributes such as operational complexity integration, cross-system interoperability, robust security, modular architecture, and user-friendly accessibility form the backbone of effective tokenization systems. These components enable seamless functionality, ensure compliance, and promote scalability. By understanding and implementing these key properties, tokenized financial solutions can achieve resilience, innovation, and efficiency, meeting the demands of today's dynamic market and paving the way for future advancements.

1.1 Sophistication and Adaptability

Operational Complexity Integration
Tokenization systems must be capable of integrating complex operational logics and specific business requirements. This includes managing financial instruments throughout their entire lifecycle, from issuance to maturity.

These systems need to handle advanced processes such as asset creation, transaction processing, compliance monitoring, and settlement, ensuring that they can support a wide range of financial products and services.

Functional Adaptability
These systems should demonstrate high levels of functional sophistication and adaptability.

They need to operate seamlessly across various environments and blockchains, facilitating interactions between tokenized assets and instruments within different systems and platforms, including traditional financial markets.

1.2 Interoperability and Compatibility

Cross-System Functionality
Interoperability is paramount. Tokenization systems should ensure compatibility and seamless functionality across different blockchain networks, whether public or private. This cross-system functionality must also extend to traditional financial systems to ensure efficient data harmonization.

Standardization and Data Alignment
Effective tokenization systems must adhere to standardized data structures and protocols. Standardization is crucial for achieving interoperability, allowing different systems and platforms to work together seamlessly. Adhering to common standards ensures that data can be accurately and efficiently exchanged between various components, reducing the risk of errors and discrepancies.

1.3 Security and Compliance

Auditability and Verifiability
Security is a non-negotiable attribute. Tokenization systems must be designed to be auditable and verifiable from a security standpoint, ensuring robust operational integrity. This means incorporating practices and features that allow for comprehensive security verification, tracking and logging of all operations within the system. Such features enable independent audits and reviews, which are essential for identifying and mitigating potential security risks.

Operational Resilience
Ensuring operational resilience is crucial for maintaining the functionality and reliability of tokenization systems. This involves implementing robust disaster recovery plans, redundancy measures, and mitigate potential system failures.

Regulatory Compliance
Tokenization systems should comply with regulatory requirements in a verifiable manner. This includes adherence to legal standards and the ability to undergo regulatory audits without compromising system integrity.

Compliance features must be built into the system to ensure that it meets the necessary legal and regulatory frameworks, including KYC (Know Your Customer), AML (Anti-Money Laundering), and data protection laws. These systems should facilitate easy and transparent regulatory reporting, ensuring that they can demonstrate compliance to regulators at any time.

1.4 Modularity and Flexibility

Scalable Architecture
Tokenization systems need a modular architecture that supports scalability. This includes the ability to update and enhance specific modules without disrupting the entire system, allowing precise management and understanding of each component.

Component Integration
The system should facilitate the development and integration of new components, leveraging existing, tested, and validated modules. This modular approach promotes efficiency and economies of scale.

1.5 Accessibility and Operational Simplicity

Cost Reduction
Reducing development and maintenance costs is crucial. Tokenization systems should aim to progressively lower the barriers to entry, making the technology accessible to a broader range of users.

User-Friendly Design
Ensuring an intuitive and user-friendly interface is essential. This applies to both internal users, such as developers and administrators, and external users.

Open-Source Ecosystem
A strong open-source ecosystem is beneficial. It provides diverse options for all functional components and modules, encouraging innovation and collaboration within the community.

1.6 Technical and Operational Excellence

Programming Languages and Technologies
Tokenization systems should be built using widely adopted programming languages and technologies within the blockchain ecosystem, avoiding extreme niches. This ensures compatibility and ease of integration with other systems and tools.

Team Training and Specialization
Internal teams responsible for managing tokenization systems must be adequately trained and specialized. This includes providing the necessary tools and methodologies for development, product design, lifecycle management, and security.

Rapid and Agile Deployment
Speedy deployment of tokenization systems is vital for maintaining a competitive edge. This rapid rollout should benefit the associated ecosystem, whether local, regional, or global.

User Experience
Ensuring a seamless user experience is critical. Tokenization systems should mask technical complexities, providing an intuitive interface for both internal and external end-users.

1.7 Ecosystem Engagement

Stakeholder Involvement
Sustaining the engagement of key stakeholders at scale is essential for the development and durability of tokenization systems. This includes continuous interaction with regulators, financial institutions, and other critical ecosystem players.

Collaborative Development
Promoting collaborative development within the ecosystem enhances resilience and trust. Tokenization systems should support the customization and extension of applications using both private and community components.

2. Engineering Scalable Tokenization Systems

Each aspect described in section 1, collectively referred to as **functional scalability**, is critical to realizing the promise of tokenization. Large financial institutions require highly sophisticated applications that are secure and performant to the highest standards. Their use cases demand high degrees of flexibility, as each may necessitate a different protocol. Many institutional players have expressed frustration with the current approach to building tokenization capabilities, noting that while the initial investment is high, the efficiency benefits only materialize at scale.

Many aspects of functional scalability must be built directly into each of the smart contracts from the ground up. One may also hope for a tool or plug-in that can be added to achieve functional scalability. However, the nature of blockchain means that functional scalability must be native to smart contract architectures. Hence, functional scalability is best taken into account from the very beginning.

With these considerations in mind, we introduce the industry's first open-source, package-oriented framework for smart contract architectures, developed by FeverTokens. This framework aims to empower builders with fast and accessible solutions to create functionally scalable tokenization systems.

We will describe the core rationale and specifications of the framework, which is progressively being adopted by prominent industry players, and showcase a few open examples of its use in the industry.

2.1 Package-Oriented Smart Contract Framework

To build functional scalability at high speed and low cost, as we have explained previously, and for the growth of a vibrant builder-developer community ecosystem, as we shall outline in the following chapters, a package-oriented framework is key.

Drawing on package-oriented software frameworks, a package-oriented framework for smart contracts extends the design that emphasizes the use of packages or modules as functional organizers of code. It similarly entails key characteristics such as modularity, dependency management, code reuse, collaboration, and versioning, etc.

In this section, we explore how FeverTokens' open-source, package-oriented framework* establishes a foundation for ecosystem players and builders. We will explain its operation, emphasize its qualities for functional scalability, and discuss the central role of the open Diamond Standard in its design.

*https://github.com/FeverTokens/ft-package-oriented-framework

Rationale and Principles of an Open, Scalable, and Non-Restrictive Smart Contract Framework

In order for a package-oriented smart contract framework to gain mass adoption, it has to be designed with certain rules and standards that optimally enforces composability; it should also provide toolings for builders to leverage expertise in the ecosystem so as to achieve functional scalability quickly and at low cost.

Packages in such a framework can encompass a wide range of components, including on-chain elements, off-chain oracles, and multi-chain components, etc. Packages can be made either private or public. Private packages can be developed by internal teams of the project builder or by an external contractor. Public packages can also be made by either internal or external developers; the difference lies in the choice to make them publicly available to future builders. A valid option is to make public packages open-source, too. This is usually advantageous when there is strategic value in setting a popular standard.

As packages can be sourced either internally or externally, the framework must come with precise guidelines on smart contract organization, documentation, and specification for developers to follow. They should offer monetization opportunities for external providers and/or the developers of packages under the framework. Indeed, a smart contract hub can be one central piece in building the ecosystem for the framework. Such a hub functions as a global marketplace of packages.

2.2 Key Functional Considerations

First, the framework implements **modularity**. It breaks down the smart contract code into smaller, more manageable packages, which in turn consist of (often composable and reusable) smart contracts. These packages and smart contracts are independently maintainable and upgradable. An additional benefit of modularity is the ease with which codes can be organized, which has broad community appeal.

Second, **dependency management** is built into the framework. When the builder includes a package in their system, packages and smart contracts on which the package depends are automatically included and their compatibility checked. Moreover, it allows stakeholder to access tooling solutions that will help developers follow standards set out by the framework, ensuring proper functioning and interoperability.

Third, a concurrent benefit of the package-oriented framework is **package reuse**. Combined with composability, it allows builders and developers to cut development time and cost dramatically by reusing components based on functions and resources.

Fourth, by organizing tokenization systems into packages, different development teams can more easily collaborate on the same project.

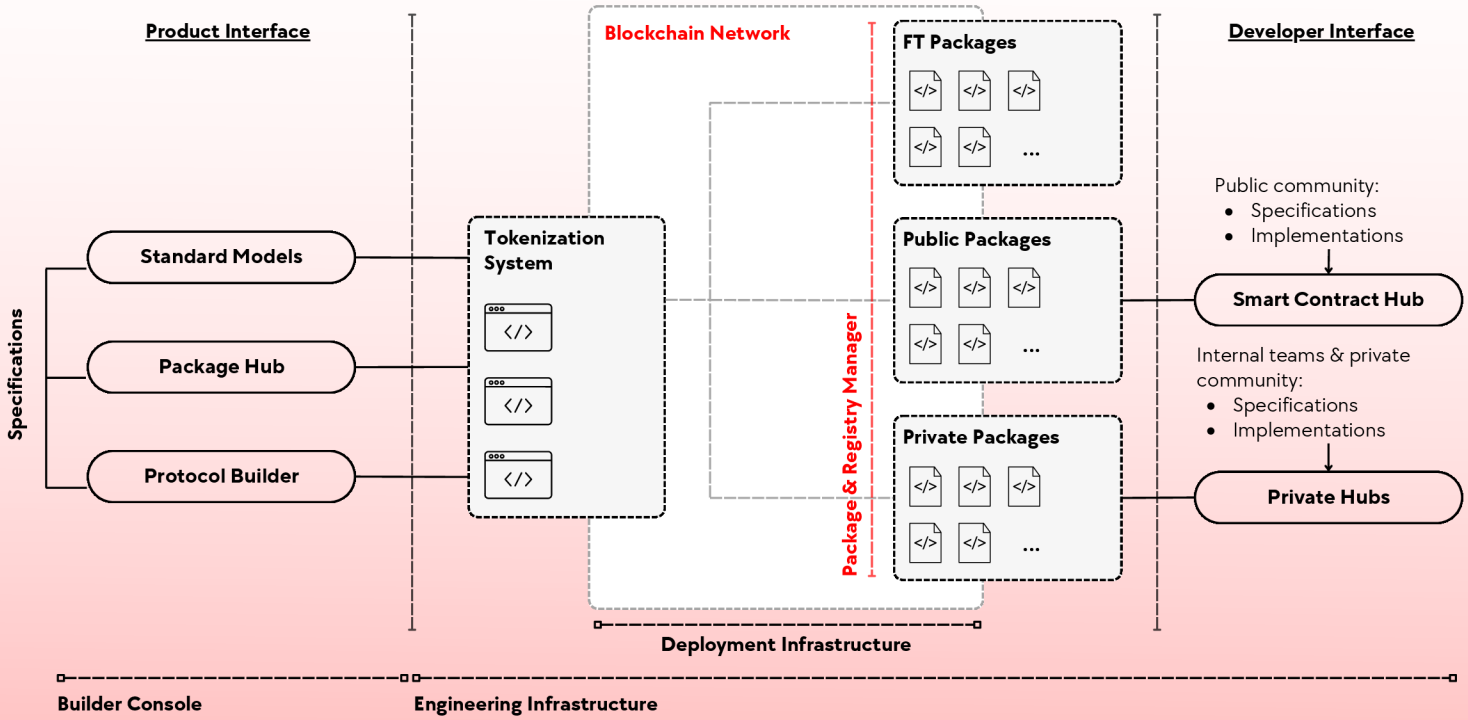
As packages follow pre-defined standards, there is little risk for incompatibility or interference as apps are developed and maintained throughout their lifecycle.

Fifth, with **version control**, the package-oriented framework further facilitates the **testing and maintenance** of new features. Packages can be updated and tested in isolation, which simplifies the management of a global tokenization system and virtually eliminates unexpected downtime.

Finally, the use of such a framework enhances **operational resilience**. In the event of bugs or hacks, it is possible to contain the issue at the package level, freeze it, and replace it according to the established governance mechanism. This avoids the need for heavy processes that could impact users.

With a package-oriented framework, internal development teams of the project builder are not "fighting" on their own. Their efforts are facilitated by not only third-party developers but also other builders in the ecosystem. The architectural design and standard enforcement of the framework ensure high flexibility, modularity, interoperability, composability, and operational resilience.

In practice, what makes the package-oriented approach so interesting, especially for large corporations, is the ability (a) to leverage the vibrant ecosystem, (b) to combine public and private smart contract hubs to achieve functional scalability, and (c) to realize and to manage sophisticated applications entirely through easy tools (no-code or low-code).



The central design consideration for the package-oriented framework is the smart engineering based on the Diamond Standard (EIP-2535). The idea is to divide features across multiple smart contracts called facets.

Central Role of the Diamond Standard

The central design consideration for such open, package-oriented smart contract framework is the smart engineering based on the Diamond Standard. These engineering efforts underpin the logic of eventual smart contract engines, hubs, and protocol building tools with community support.

The Diamond Standard is an innovative smart contract architecture. The idea is to divide features across multiple smart contracts called facets. By doing so, developers have more flexibility to add, remove, or upgrade functionalities to their application by wrapping a set of specific functions into a facet that will be invoked by a proxy.

Summarized in Ethereum Improvement Proposal-2535 (EIP-2535), the Diamond Standard provides a development pattern for modularity and flexibility. However, it alone is insufficient for the proper functioning of a scalable package-oriented framework: Specifically, the Diamond Standard enables developers to upgrade smart contract systems after deployment and overcome the size limit for EVM compilers. To achieve functional scalability in doing so, the framework needs additional heavy engineering and standardization.

Authors

ZAKARYAE BOUDI

CEO
FeverTokens
Tokenized Economies Institute

JIULIN TENG

Head of Growth
FeverTokens
Tokenized Economies Institute

MOHAMED TOUB

CTO
FeverTokens
Tokenized Economies Institute

SARA KAWAS

Project Manager
FeverTokens
Tokenized Economies Institute

NADIA FILALI

Head of Innovation and Development
Caisse des Dépôts