

세이프(Safe)

해당 번역본은 이용자가 디지털자산 관련 정보를 편리하게 확인할 수 있도록 단순 참고용으로 번역된 것이며, 투자 권유를 목적으로 하지 않습니다.

그 내용이 정확하지 않을 수 있으므로 정확한 정보 습득을 위해서는 원문을 참고하시거나 원문 작성 주체 측에 문의하시기 바랍니다.

두나무는 해당 번역본 내용의 진실성 및 정확성을 보장하지 않고, 제공된 정보에 의한 투자 결과에 대하여 어떠한 책임도 지지 않으며, 투자 손실은 투자자에게 귀속됩니다.

Safe{Core} 프로토콜

초안

루카스 쇼어(Lukas Schor), 리처드 마이스너(Richard Meissner)

2023년 8월

초록 (ABSTRACT)

Safe{Core} 프로토콜은 모듈형 스마트 컨트랙트 기반 계정(“스마트 계정”)을 위한 오픈소스 상호운용성 프로토콜을 제안한다. 이는 스마트 계정 생태계에서 발생하는 **단편화**, **벤더 종속**, **보안 위험**과 같은 문제를 해결한다. 이 벤더 중립적인 프로토콜은 **플러그인(Plugins)**, **훅(Hooks)**, **서명 검증기(Signature Validators)**, **함수 처리기(Function Handlers)** 등과 같은 **모듈(Modules)**을 지원하여 **계정(Accounts)**의 기능을 확장할 수 있게 하며, **레지스트리(Registries)**는 표준 및 보안 속성의 준수를 보장한다. 향후 버전에서는 체인 추상화, 수수료 메커니즘, 새로운 모듈 유형 지원 등이 도입될 예정이다. Safe{Core} 프로토콜은 조합성(composability), 이식성(portability), 보안을 확보함으로써 스마트 계정의 장기적 성장과 발전을 촉진하기 위한 생태계 주도의 이니셔티브이다.

1. 서론 (INTRODUCTION)

1.1. 계정 추상화 (Account Abstraction)

이더리움 가상 머신(Ethereum Virtual Machine, EVM) 생태계가 발전하기 위해서는, 외부 소유 계정(Externally Owned Account, EOA)에서 **스마트 계정(Smart Account)**으로의 *전환(transition)*이 필요하다. 이를 통해 복구 메커니즘, 다중서명 지갑, 트랜잭션 배치, 자동화, 가스 추상화 등과 같은 사용성과 보안 기능을 활성화할 수 있다. 이러한 전환은 일반적으로 *계정 추상화(Account Abstraction)*라 불리며, 다음과 같은 다양한 표준화 노력에 의해 지원되고 있다:

- **ERC-1271**은 스마트 계정이 오프체인 기능을 처리할 수 있도록 설계되었다. 이 표준은 서명 검증을 위한 견고하고 사용자 정의 가능한 방법을 제공하여 스마트 계정의 범용성을 강화한다.
- **ERC-4337**은 컨센서스 레이어를 변경하지 않고 계정 기능을 표준화하고 추상화하려는 시도이다. 이 표준의 주요 목표는 스마트 계정이 임의의 트랜잭션 수수료 지불 메커니즘을 구현할 수 있도록 하는 것이다.
- **ERC-6900**은 공통 인터페이스를 통해 스마트 계정의 모듈화 문제를 해결하고, 효율적이며 재사용 가능한 스마트 계정 모듈의 기반을 조성한다.

이러한 시도들(**EIP-3074, ERC-5005, ERC-6492, EIP-7377** 등 포함)은 각기 다른 스마트 계정의 특정 측면을 다루는 **구성 요소적 빌딩 블록**이지만, 지금까지는 개념적으로만 연결되어 있었다.

Safe{Core} 프로토콜은 이러한 이니셔티브를 명시적으로 통합함으로써, 스마트 계정 개발을 간소화하고 스마트 계정 생태계의 **광범위한 채택과 확장**을 촉진하고자 한다.

1.2. 동기 (Motivation)

Safe{Core} 프로토콜은 조합 가능하고(composable), 이식 가능하며(portable), 안전한(secure) **모듈형 스마트 계정**을 위한 오픈소스 프레임워크이다. 이 프로토콜은 *Safe Smart Account*에서 얻은 경험을 바탕으로 하지만, 근본적으로 **벤더 중립적(vendor-agnostic)**이며 **생태계 전체를 위한 이니셔티브**로서 다음의 문제들을 해결하고자 한다:

- **단편화(Fragmentation)**: 다양한 벤더가 각자의 스마트 계정을 개발함에 따라, 툴링과 애플리케이션은 특정 구현과의 호환성을 우선시해야 하는 상황이 발생한다. 그 결과 스마트 계정 생태계의 심각한 단편화와 구성 요소 재사용의 부족이 초래될 수 있다. 이러한 개발자 및 사용자 경험의 저하로 인해 스마트 계정의 전반적 성장과 활용성이 감소한다.
- **벤더 종속(Vendor Lock-In)**: 외부 소유 계정(EOA)은 지갑과 애플리케이션 간의 이식성을 보장하는 널리 수용된 표준이다. 반면, 스마트 계정의 단편화는 각 지갑이 고유한 계정 구현체를 만들면서 이식성을 저해하고, 결과적으로 벤더 종속을 유발할 수 있다. 사용자는 자산 이동이나 다른 서비스와의 상호작용 시 제약을 겪게 된다.
- **보안 위험(Security Risks)**: 스마트 계정은 브리지나 디파이(DeFi) 프로토콜과 같은 다른 스마트 컨트랙트 기반 시스템과 유사한 **스마트 컨트랙트 리스크**를 도입한다. 충분한 보안이 확보되지 않으면, 사용자는 스마트 계정의 잠재력을 신뢰하고 활용하기 어렵다.

이러한 문제를 해결하기 위해 *Safe{Core}* 프로토콜은 구성 요소의 **최대 재사용성과 상호운용성**, 계정의 **이식성 유지**, 그리고 **강화된 보안 속성**의 확립을 목표로 한다. *Safe{Core}* 프로토콜을 채택하는 개발자는 **유연성(flexibility)**, **상호운용성(interoperability)**, **조합성(composability)**을 확보할 수 있으며, 향후에는 수수료 부과를 위한 기능적 기반도 제공받게 된다.

2. 프로토콜 구성 요소 (PROTOCOL COMPONENTS)

Safe{Core} 프로토콜의 아키텍처는 각각 고유한 책임과 인터페이스를 가진 여러 구성 요소로 이루어져 있다. 이러한 구조는 각 구성 요소의 독립적인 확장과 반복(iteration)을 가능하게 하며, **모듈성(modularity)**을 통해 스마트 계정 생태계의 지속 가능한 성장과 진화를 보장한다.

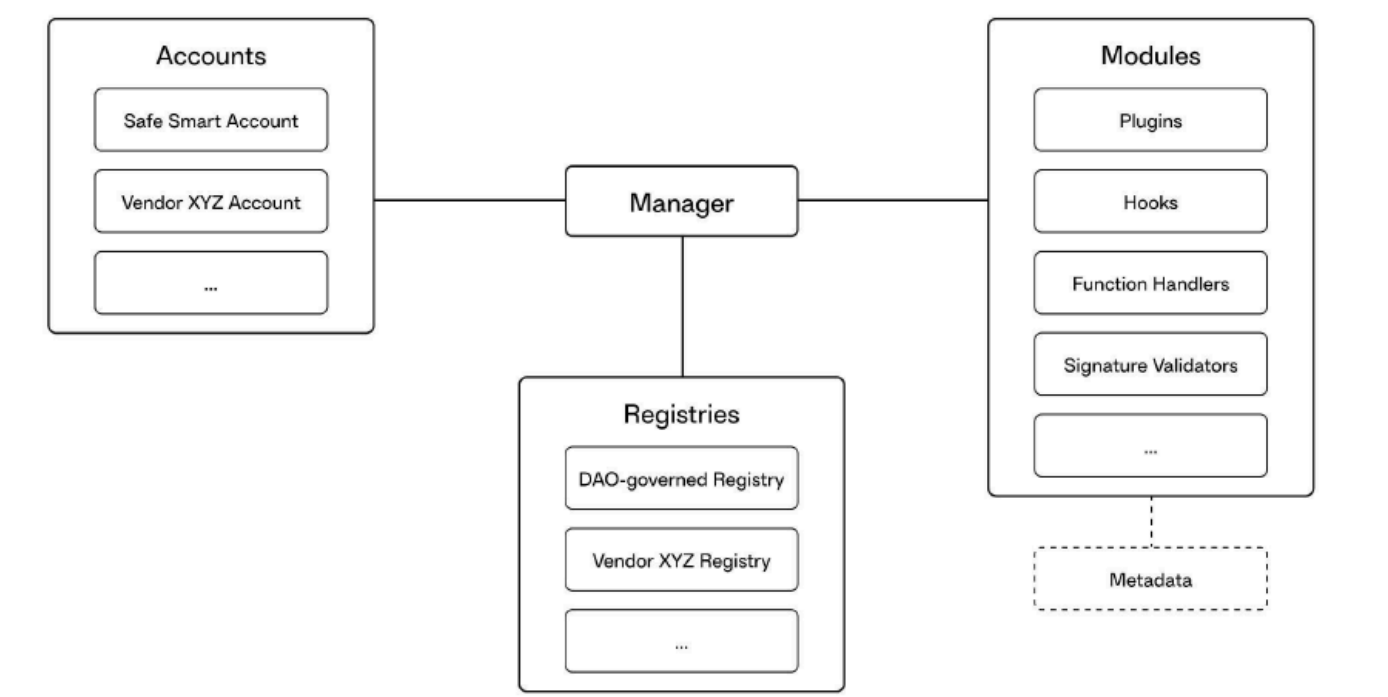


그림 1: Safe{Core} 프로토콜 아키텍처

2.1. 매니저 (Manager)

프로토콜의 중심에는 매니저(Manager)가 있으며, 이는 계정(Accounts), 레지스트리(Registries), 모듈(Modules) 간의 조정 지점으로 작동한다(그림 1 참조). 매니저는 스마트 계정의 모듈성을 처리하기 위한 추상화 계층을 구현한다. 사용자는 선택한 레지스트리 내에 등록된 모듈을 활성화하여 계정의 기능을 확장할 수 있다(그림 2 참조).

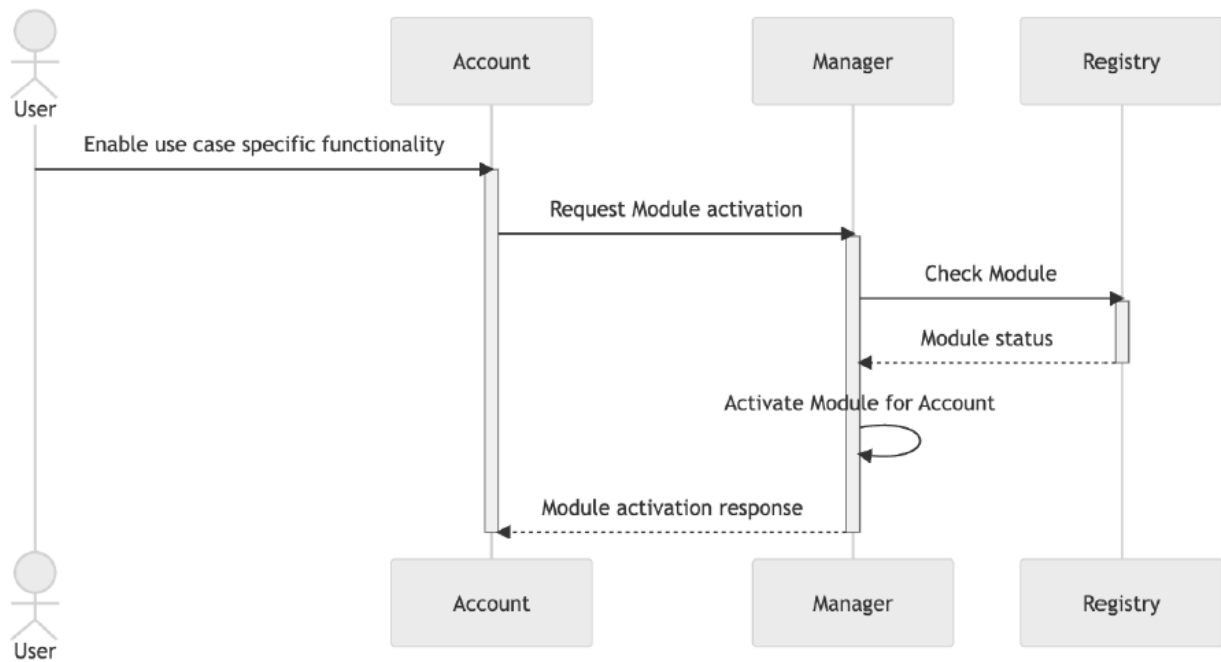


그림 2: 모듈 활성화 흐름(Module activation flow)

매니저는 또한 권한 부여(permissioning) 계층으로 작동함으로써 스마트 계정의 보안을 확장한다. 이를 통해 서로 다른 모듈에 대해 다양한 수준의 권한을 부여할 수 있으며, 예를 들어 모듈이 계정에 대해 구성 변경이나 `delegatecall` 연산을 수행하도록 제한할 수 있다. 또한, 매니저는 네트워크에 따라 달라질 수 있는 다양한 기본 트랜잭션 수수료 처리 체계(예: ERC-4337)를 구현할 수 있도록 지원한다.

매니저는 여러 계정 간에 공유되며 사용자가 직접 활성화한다. 보안상의 이유로 업그레이드가 불가능하지만, 시간이 지남에 따라 새로운 기능이 추가된 배포가 이루어질 것으로 예상되며, 사용자가 스스로 마이그레이션할 수 있도록 설계되어 있다.

2.2. 계정 (Accounts)

계정(Accounts)은 사용자가 소유한 스마트 계정으로, 매니저(Manager)를 활성화함으로써 프로토콜에 옵트인(opt-in)한다. 이 활성화는 계정 배포 시점에서 이루어질 수도 있고, 첫 번째 모듈을 추가할 때 발생할 수도 있다. Safe{Core} 프로토콜은 **벤더 중립적(vendor-agnostic)**이므로, 매니저가 정의한 계정 인터페이스를 지원하는 한 특정 계정 구현체에 종속되지 않는다.

초기 계정 인터페이스는 개발 속도를 높이고 피드백을 수집하기 위해 Safe Smart Account의 Lx 버전에 최적화되어 있다. 이러한 경험은 계정 인터페이스의 일반화(generalization)를 위한 기반이 될 것이며, 이를 통해 기존 및 신규 계정 구현체가 프로토콜과 호환될 수 있도록 하고, 생태계를 위한 개방형 인프라 구조를 제공한다.

2.3. 모듈 (Modules)

모듈(Modules)은 다양한 방식으로 계정(Accounts)의 기능을 확장한다. 초기 모듈 유형에는 플러그인(Plugins), 훅(Hooks), 서명 검증기(Signature Validators), 함수 처리기(Function Handlers)가 포함되며, 시간이 지남에 따라 Safe{Core} 프로토콜에 새로운 유형이 추가될 수 있다. 이러한 모듈 유형은 ERC-6900과 같은 표준의 영향을 받는 것으로 가정된다.

2.3.1. 플러그인 (Plugins)

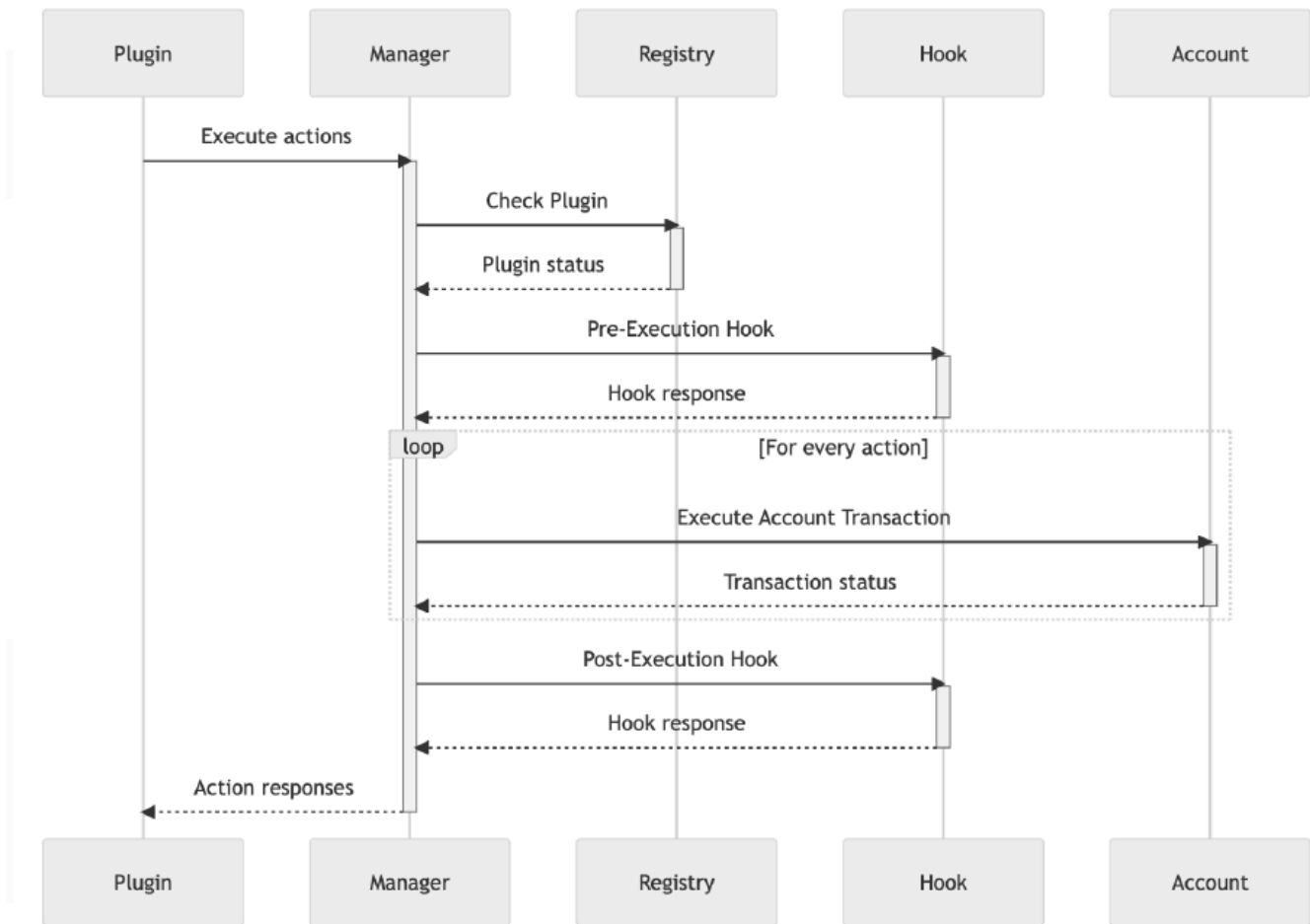
플러그인(Plugins)은 복구 메커니즘, 세션 키, 자동화 등 임의의 로직을 계정에 추가할 수 있도록 한다. 이러한 플러그인은 외부에 배포된 스마트 컨트랙트 형태로 존재하며, 메인/자에서 활성화되어 계정 트랜잭션을 트리거하는 대체 진입점으로 작동한다(그림 3 참조). 이들은 지갑에 추가적인 기능을 제공할 수 있을 뿐 아니라, 개발자가 더 강력한 애플리케이션을 구축할 수 있도록 하는 앱 전용 플러그인으로도 활용될 수 있다.

2.3.2. 훅 (Hooks)

훅(Hooks)은 트랜잭션 생명주기의 특정 시점에서 추가 로직을 삽입한다. Safe{Core} 프로토콜은 두 가지 유형의 훅을 정의한다:

- 사전 실행 훅(Pre-Execution Hooks): 트랜잭션이 실행되기 전에 작동한다.
- 사후 실행 훅(Post-Execution Hooks): 트랜잭션이 종료된 후 작동한다.

훅은 허용/차단 목록, MEV 및 슬리피지 방어, 리스크 및 컴플라이언스 평가 등 다양한 형태의 보안 보호 메커니즘을 구현할 수 있게 한다.



2.3.3. 서명 검증기 (Signature Validators)

서명 검증기(Signature Validators)는 **ERC-1271** 표준을 활용하여 (도메인 기반) 서명 검증을 위한 사용자 정의 가능한 로직을 구현할 수 있게 한다. 서명 검증기는 오프체인 기능을 통합함으로써 스마트 계정의 범용성을 높인다. 또한, 이는 CowSwap의 **시간가중평균가격(Time-Weighted Average Price, TWAP)** 기능과 같은 **의도 기반 아키텍처(intent-based architecture)**를 구현할 수 있도록 한다.

2.3.4. 함수 처리기 (Function Handlers)

함수 처리기(Function Handlers)는 **계정(Account)** 내에서 지원되지 않는 함수가 호출될 때 임의의 함수 호출을 관리한다. 이를 통해 **계정**은 최소한의 로직만을 포함하면서도 새로운 토큰 유형 등 신흥 표준과의 호환성을 유지할 수 있다. 다른 모든 모듈 유형과 마찬가지로, **함수 처리기** 역시 주의 깊게 활성화해야 한다. 왜냐하면 읽기 전용 함수조차 외부 컨트랙트가 이를 인증(authentication)에 의존할 경우 **계정**의 상태에 영향을 미칠 수 있기 때문이다.

2.4. 레지스트리 (Registries)

레지스트리(Registries)는 **모듈(Modules)**의 **보안 속성을 강화**하고 **표준(예: ERC-6900)**을 강제하는데 핵심적인 역할을 한다. 이들의 역할은 승인되고 사용 가능한 **모듈**의 포괄적인 목록을 보유한 **중앙 저장소** 또는 **스마트 계정용 앱스토어**와 유사하다. **계정(Account)**은 **매니저(Manager)**에서 사용할 레지스트리를 설정하며, 그 결과 해당 레지스트리에 승인된 **모듈**만 활성화할 수 있다.

보안을 강화하기 위해, **레지스트리**는 고유한 포함/제외 기준 및 절차를 설정할 수 있다. 예를 들어, 특정 **레지스트리**는 취약점을 자동으로 탐지하고, 문제 있는 **모듈**을 **계정**에서 자동으로 비활성화하는 프로세스를 구축할 수 있다. 이 경우 사용자는 일정한 유예 기간(grace period) 내에 이 결정을 무효화할 수도 있다. 이러한 메커니즘을 통해 사용자는 **모듈**의 기능 확장성을 누리면서도 높은 보안 표준을 유지할 수 있다.

또한 **레지스트리**는 큐레이션(curated) 방식을 통해 검색성을 향상시킬 수 있으며, 개발자와 사용자에게 **참조 포인트(reference point)** 역할을 수행한다. 예를 들어, 특정 **레지스트리**는 특정 지갑 벤더의 사용자 인터페이스(UI)에서 지원되는 **모듈**만 노출함으로써 더 강력한 보안 및 사용성 속성을 보장할 수 있다.

2.5. 메타데이터 (Metadata)

Safe{Core} 프로토콜 위에 강력한 툴링, 지갑, 애플리케이션을 구축하기 위해 개발자는 **모듈** 및 모듈 트랜잭션에 대한 추가적인 컨텍스트를 **메타데이터(Metadata)** 형태로 제공할 수 있다. 이를 통해 개발자는 “이름(name)”, “버전(version)”, “설명(description)” 등과 같은 추가 정보를 첨부할 수 있다. 예를 들어, 개발자는 **모듈**의 사용자 인터페이스로 활용 가능한 프론트엔드 위젯 링크를 포함할 수 있다.

또한 **레지스트리**는 **모듈**의 버전 정보를 활용하여 사용자에게 업데이트를 제안할 수도 있다. **메타데이터**는 특정 트랜잭션이 어떤 지갑 벤더를 통해 수행되었는지와 같은 추가적인 맥락 정보를 제공하는 데도 활용될 수 있다.

메타데이터는 ERC-712를 활용하여 정보를 기계 판독 가능(machine-readable), 온체인 검증 가능(on-chain-verifiable), 그리고 부분적으로 사람이 읽을 수 있는(semi-human-readable) 형태로 만든다.

3. 전망 (OUTLOOK)

위 내용은 Safe{Core} 프로토콜의 초기 개념을 구성하며, 현재 초기 참조 구현(reference implementation)이 진행 중이다. 이 프로토콜은 향후 여러 방향으로 발전할 수 있는 여지를 가지고 있다.

3.1. 수수료 메커니즘 (Fees Mechanisms)

계정(Accounts), 레지스트리(Registries), 모듈(Modules)은 일회성 활성화 수수료, 사용량 기반 수수료, 구독형 수수료 등 다양한 형태의 수수료를 부과할 수 있어야 한다. 이러한 수수료는 개발자가 직접 수취할 수도 있고, 지갑 제공자나 레지스트리 유지 관리자 등 다른 이해관계자와 공유될 수도 있다. 이는 Safe{Core} 프로토콜에 참여하고 기여하는 데 있어 재정적 인센티브를 제공함으로써, 보다 지속 가능하고 활기찬 스마트 계정 생태계를 조성할 수 있다.

3.2. 체인 추상화 (Chain Abstraction)

현재의 아키텍처에서는 Safe{Core} 프로토콜 및 그 구성 요소를 각 체인마다 개별적으로 배포해야 한다. 향후 버전에서는 프로토콜 구성 요소를 신뢰 최소화(trust-minimized) 또는 완전 신뢰 불필요(trustless) 방식으로 체인 간(cross-chain) 보다 쉽게 접근할 수 있는 방식을 도입해야 한다.

3.3. 추가 모듈 유형 (More Module Types)

기존 모듈(Modules) — 플러그인(Plugins), 훅(Hooks), 서명 검증기(Signature Validators), 함수 처리기(Function Handlers) — 외에도 스마트 계정의 유연성과 기능을 확장할 수 있는 새로운 모듈 유형이 Safe{Core} 프로토콜에 추가될 수 있다. 또한 기존 모듈 유형 역시 확장될 수 있으며, 예를 들어 훅(Hooks)을 플러그인 전용 훅을 지원하도록 확장할 수 있다.

4. 결론 (CONCLUSION)

Safe{Core} 프로토콜은 스마트 계정 개발을 진전시키기 위한 모듈형 오픈소스 상호운용성 프로토콜의 선구적 사례로, 기능 확장을 위한 포괄적 모듈 유형 세트를 도입한다. 그 벤더 중립적 설계는 구성 요소의 높은 재사용성을 보장하고 계정의 이식성을 유지한다. 또한 레지스트리(Registries)는 모듈(Modules)의 검색 가능성을 제공하고, 보안 및 상호운용성 속성을 유지하며, 취약성 대응을 촉진한다. 앞으로 프로토콜은 체인 추상화를 수용하고, 수수료 구조를 확립하며, 추가적인 모듈 유형을 통합하는 방향으로 진화할 것이다. 이 연구는 Safe{Core} 프로토콜이 EVM 생태계의 스마트 계정 전환을 지원하고, 개발자 및 사용자 경험을 개선할 잠재력을 보여준다.