

유니스왑(Uniswap)

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.

Uniswap v4 핵심

2024년 8월

헤이든 애덤스 (Hayden Adams) hayden@uniswap.org	무디 세일럼 (Moody Salem) moody.salem@gmail.com	노아 진스마이스터 (Noah Zinsmeister) noah@uniswap.org
사라 레이놀즈 (Sara Reynolds) sara@uniswap.org	오스틴 애덤스 (Austin Adams) me@aada.ms	윌 포트 (Will Pote) willpote@gmail.com
마크 토다 (Mark Toda) mark@uniswap.org	앨리스 헨쇼 (Alice Henshaw) alice@uniswap.org	에밀리 윌리엄스 (Emily Williams) emily@uniswap.org
댄 로빈슨 (Dan Robinson) dan@paradigm.xyz		

초록 (ABSTRACT)

UNISWAP v4는 Ethereum Virtual Machine(EVM)을 위한 비(非)커스터디형 자동화 시장조성자 (AMM)입니다.

UNISWAP v4는 임의의 코드 후크(hook)를 통한 커스터마이징 기능을 제공하여, 개발자가 UNISWAP v3에서 도입된 집중형 유동성(concentrated liquidity) 모델에 새로운 기능을 추가할 수 있도록 합니다.

UNISWAP v4에서는 누구나 사전 정의된 풀 동작 전후에 실행되는 후크를 지정하여 새로운 풀을 생성할 수 있습니다.

후크는 이전에는 프로토콜에 내장되어 있던 기능(예: 오라클)뿐만 아니라, 이전에는 독립적인 프로토콜 구현이 필요했던 새로운 기능들도 구현할 수 있도록 합니다.

UNISWAP v4는 또한 싱글톤(singleton) 구현, 플래시 회계(flash accounting), 네이티브 ETH 지원을 통해 가스 효율성과 개발자 경험을 개선합니다.

1 서론 (INTRODUCTION)

UNISWAP v4는 Ethereum Virtual Machine(EVM) 상에서의 효율적인 가치 교환을 지원하는 자동화 시장조성자(AMM)입니다.

기존 UNISWAP 프로토콜의 버전들과 마찬가지로, 이 프로토콜은 비(非)커스터디형이며, 업그레이드 불가능하고, 퍼미션리스(permissionless)입니다.

UNISWAP v4는 개발자를 위한 추가적인 커스터마이징과 가스 효율성을 위한 아키텍처 변경에 중점을 두고 있으며,

UNISWAP v1 및 v2에서 구축된 AMM 모델과 UNISWAP v3에서 도입된 집중형 유동성 모델을 기반으로 합니다.

UNISWAP v1 [2]과 v2 [3]는 UNISWAP 프로토콜의 초기 두 가지 버전으로, 각각 ERC-20 ↔ ETH, ERC-20 ↔ ERC-20 스왑을 가능하게 하였으며,

모두 상수곱 자동시장조성자(Constant Product Market Maker, CPMM) 모델을 사용하였습니다.

UNISWAP v3 [4]는 집중형 유동성을 도입하여, 제한된 가격 범위 내에서 유동성을 제공하는 포지션을 통해 자본 효율적인 유동성 공급을 가능하게 하였고, 여러 개의 수수료 계층(fee tier)을 지원하였습니다.

집중형 유동성과 수수료 계층은 유동성 공급자에게 유연성을 제공하고, 새로운 유동성 제공 전략을 가능하게 했지만,

UNISWAP v3는 AMM과 DeFi가 발전함에 따라 등장한 새로운 기능들을 지원할 유연성이 부족합니다.

예를 들어, UNISWAP v2에서 도입되어 v3에 포함된 가격 오라클과 같은 기능은

통합자(integrator)가 탈중앙 온체인 가격 데이터를 활용할 수 있도록 해주지만,

스왑 사용자에게는 높은 가스 비용을 야기하며, 통합자에게는 커스터마이징이 불가능합니다.

또한, 시간가중 평균 가격 주문(TWAP)을 지원하는 시간가중 평균 시장조성자(TWAMM) [8],

변동성 오라클, 지정가 주문, 동적 수수료 등의 기능은 프로토콜의 핵심을 재구현해야만 가능하며,

UNISWAP v3에는 서드파티 개발자가 이러한 기능을 추가할 수 없습니다.

더불어, 기존 UNISWAP 버전에서는 새로운 풀을 배포할 때마다 새로운 컨트랙트를 배포해야 했고,

이는 바이트코드의 크기에 비례하여 비용이 증가했습니다.

또한, 여러 UNISWAP 풀 간의 거래는 다수의 컨트랙트 간 전송과 중복된 상태 업데이트를 수반하였습니다.

그리고 UNISWAP v2부터는 네이티브 ETH 대신 ERC-20 형태의 래핑된 WETH를 사용해야 했습니다.

이러한 설계 선택은 최종 사용자에게 추가적인 가스 비용을 초래했습니다.

UNISWAP v4에서는 다음과 같은 몇 가지 주요 기능을 통해 기존의 비효율성을 개선합니다:

- **후크(Hooks):** UNISWAP v4는 누구나 맞춤 기능을 가진 새로운 집중형 유동성 풀을 배포할 수 있도록 합니다.
각 풀마다 생성자는 호출의 생애주기(lifecycle) 중 특정 지점에서 실행되는 로직을 구현한 “후크 컨트랙트(hook contract)”를 정의할 수 있습니다.
이러한 후크는 풀의 스왑 수수료를 동적으로 관리하거나, 사용자 정의 커브(custom curve)를 구현하고,
사용자 정의 회계(Custom Accounting)를 통해 유동성 공급자 및 스왑 사용자에게 부과되는 수수료를 조정할 수 있습니다.
- **싱글톤(Singleton):** UNISWAP v4는 이전 버전에서 사용된 팩토리(factory) 모델을 폐기하고, 모든 풀을 하나의 컨트랙트에 포함시키는 단일 계약 구조(singleton)를 채택합니다.
이 모델은 풀 생성 비용과 다중 홉(multi-hop) 거래의 비용을 절감합니다.
- **플래시 회계(Flash accounting):** 싱글톤은 “플래시 회계”를 사용하며, 호출자가 락(lock) 기간 동안 풀을 잠그고 해당 풀의 모든 토큰에 접근할 수 있도록 합니다.
단, 락 종료 시점에 호출자와 풀 간에 미결제된 토큰이 없어야 합니다.
이 기능은 EIP-1153 [5]에서 설명된 일시적 저장(transient storage) 연산자를 통해 효율적으로 구현됩니다.
플래시 회계는 다중 풀을 경유하는 거래의 가스 비용을 더욱 줄여주며, UNISWAP v4와의 복잡한 통합도 지원합니다.
- **네이티브 ETH 지원(Native ETH):** UNISWAP v4는 네이티브 ETH에 대한 지원을 다시 도입하였으며,
v4 풀 내에서 네이티브 토큰과의 페어를 구성할 수 있습니다.
ETH 기반의 스왑 사용자 및 유동성 공급자는 더 저렴한 전송 비용과 래핑 제거를 통해 가스 비용 절감 혜택을 누릴 수 있습니다.
- **사용자 정의 회계(Custom Accounting):** 싱글톤은 후크가 반환하는 델타(delta)를 활용하여 기본 집중형 유동성 풀을 확장하거나 우회할 수 있습니다.
이를 통해 싱글톤을 연결된 풀의 불변 결제 계층(immutable settlement layer)으로 사용할 수 있게 됩니다.
이 기능은 후크 기반 출금 수수료, 자산 래핑, 또는 UNISWAP v2와 같은 상수급 시장조성자(CPMM) 커브 등 다양한 사용 사례를 지원할 수 있습니다.

다음 섹션에서는 이러한 기능들과 이를 가능하게 하는 아키텍처 변경 사항들에 대해 자세히 설명합니다.

2 후크(HOOKS)

후크(Hooks)는 외부에 배포된 컨트랙트로서, 풀의 실행 중 지정된 시점에 개발자가 정의한 로직을 수행합니다.

이러한 후크는 통합자가 유연하고 커스터마이징 가능한 실행을 가진 집중형 유동성 풀을 생성할 수 있도록 합니다.

선택적으로, 후크는 사용자 정의 회계(Custom Accounting) 섹션(5)에서 설명된 대로, 스왑의 동작을 변경할 수 있는 커스텀 델타(custom delta)를 반환할 수도 있습니다.

후크는 풀의 파라미터를 수정하거나 새로운 기능을 추가할 수 있습니다.

후크를 사용하여 구현할 수 있는 예시 기능은 다음과 같습니다:

- TWAMM [8]을 통해 대규모 주문을 시간에 따라 나눠서 실행
- 틱 가격에 도달했을 때 체결되는 온체인 지정가 주문
- 변동성을 반영한 동적 수수료
- 유동성 공급자를 위한 MEV 내부화 메커니즘 [1]
- 중앙값, 절삭형(truncated), 또는 기타 사용자 정의 오라클 구현
- 상수급 시장조성자(UNISWAP v2 기능과 동일)

2.1 액션 후크(Action Hooks)

UNISWAP v4에서 사용자가 풀을 생성할 때, 후크 컨트랙트를 지정할 수 있습니다.

이 후크 컨트랙트는 풀의 실행 중 호출되는 커스텀 로직을 구현합니다.

UNISWAP v4는 현재 다음과 같은 총 10개의 후크 콜백(callback)을 지원합니다:

- beforeInitialize / afterInitialize
- beforeAddLiquidity / afterAddLiquidity¹
- beforeRemoveLiquidity / afterRemoveLiquidity
- beforeSwap / afterSwap
- beforeDonate / afterDonate

어떤 후크 콜백이 실행될지는 후크 컨트랙트의 주소에 의해 결정됩니다.

이는 실행할 콜백을 효율적이고 명확하게 정의할 수 있는 방법론을 제공하며, 업그레이드 가능한 후크조차도 특정 불변 조건(invariant)을 반드시 따르도록 보장합니다.

작동하는 후크를 생성하기 위한 최소 요건은 매우 간단합니다.

Figure 1에서는 스왑 실행 흐름의 일부로서 beforeSwap 및 afterSwap 후크가 어떻게 동작하는지 설명합니다.

¹ beforeAddLiquidity와 beforeRemoveLiquidity에 별도의 권한을 부여하는 것은 두 액션 간의 보안 가정의 차이를 반영합니다. 유동성을 소각(burn)하지 않고 주조(mint)만 영향을 줄 수 있는 후크는 유동성 공급자에게 더 안전한데, 이는 그들이 언제든지 유동성을 인출할 수 있다는 보장이 있기 때문입니다.

2.2 후크가 관리하는 수수료(Hook-managed fees)

UNISWAP v4에서는 후크를 통해 스왑 시 수수료를 부과할 수 있습니다.

스왑 수수료는 고정형(static)일 수도 있고,

후크 컨트랙트에 의해 동적으로 관리될 수도 있습니다.

후크 컨트랙트는 스왑 수수료의 일부를 자신에게 할당하도록 선택할 수도 있습니다.

후크 컨트랙트에 누적되는 수수료는 해당 후크의 코드에 따라

유동성 공급자, 스왑 사용자, 후크 생성자 또는 기타 참여자 등에게 임의로 분배할 수 있습니다.

후크의 기능은 풀 생성 시 선택된 변경 불가능한 플래그(immutable flag)에 의해 제한됩니다.

예를 들어, 풀 생성자는 해당 풀이 고정 수수료를 사용할지 여부 및 수수료율을 결정하거나, 동적 수수료를 사용할 수 있도록 선택할 수 있습니다.

거버넌스 또한, 아래의 거버넌스 섹션(6.2)에서 논의된 바와 같이, 스왑 수수료의 일정 비율을 상한(cap) 내에서 가져갈 수 있습니다.

3 싱글톤 및 플래시 회계 (SINGLETON AND FLASH ACCOUNTING)

기존 UNISWAP 프로토콜 버전들은 팩토리/풀(factory/pool) 패턴을 사용하여, 팩토리가 새로운 토큰 페어마다 별도의 컨트랙트를 생성하는 구조를 따랐습니다. UNISWAP v4는 싱글톤(*singleton*) 디자인 패턴을 사용하여 모든 풀을 하나의 컨트랙트에서 관리하며, 이를 통해 풀 배포 비용이 99% 절감됩니다.

싱글톤 디자인은 v4의 또 다른 아키텍처 변화인 플래시 회계(*flash accounting*)를 보완합니다. 기존 UNISWAP 프로토콜에서는 대부분의 작업(예: 스왑 또는 유동성 추가)이 토큰 전송으로 종료되었습니다.

v4에서는 각 작업이 델타(*delta*)라고 하는 내부 순잔액(*net balance*)을 업데이트하며, 락 종료 시에만 외부 전송이 이루어집니다. 새로운 `take()`와 `settle()` 함수는 각각 풀로부터 자금을 차입하거나 예치하는 데 사용됩니다. 호출 종료 시점에 호출자와 풀 매니저 간에 미상환된 토큰이 없도록 요구함으로써, 풀의 지급 능력(*solvency*)이 강제됩니다.

플래시 회계는 아토믹 스왑 및 유동성 추가와 같은 복잡한 풀 작업을 단순화합니다. 싱글톤 모델과 결합되면, 다중 홉 스왑이나 스왑 후 유동성 추가와 같은 복합 연산도 간소화됩니다.

칸쿤 하드포크(Cancun hard fork) 이전까지는, 플래시 회계 구조가 비용이 많이 들었습니다. 왜냐하면 잔액 변경 시마다 저장소 업데이트가 필요했기 때문입니다. 비록 컨트랙트가 내부 회계 데이터를 실제로는 저장소에 직렬화하지 않는다는 것을 보장했지만, 저장소 환급 한도(*storage refund cap*)를 초과하면 사용자에게 여전히 동일한 비용이 부과되었습니다 [6].

하지만 트랜잭션 종료 시 잔액이 반드시 0이 되어야 하므로, 이러한 잔액 처리는 EIP-1153 [5]에서 정의된 일시적 저장(*transient storage*)을 통해 구현될 수 있습니다.

이처럼 싱글톤과 플래시 회계를 결합함으로써, 여러 v4 풀을 경유하는 라우팅을 보다 효율적으로 처리할 수 있으며, 유동성 분산(*liquidity fragmentation*)에 따른 비용을 줄일 수 있습니다. 이는 특히 후크(*hook*)의 도입으로 인해 풀의 수가 급증하게 되는 상황에서 매우 유용합니다.

4 네이티브 ETH (NATIVE ETH)

UNISWAP v4는 거래 페어에서 네이티브 ETH를 다시 지원합니다.
UNISWAP v1은 엄격히 ETH ↔ ERC-20 토큰 페어만 지원했지만,
UNISWAP v2에서는 구현 복잡성과 WETH/ETH 페어 간 유동성 분산 우려로 인해 네이티브 ETH 페어가 제거되었습니다.
그러나 싱글톤 및 플래시 회계 구조는 이러한 문제를 완화하므로,
UNISWAP v4에서는 WETH와 ETH 페어 모두를 허용합니다.

네이티브 ETH 전송은 ERC-20 전송보다 가스 비용이 약 절반입니다
(ETH는 21k 가스, ERC-20은 약 40k 가스 소모).
현재 UNISWAP v2와 v3에서는 대부분의 사용자가 거래 전후에 ETH를 WETH로 래핑(wrap)하거나 언래핑(unwrap)해야 하며,
이로 인해 추가적인 가스가 발생합니다.

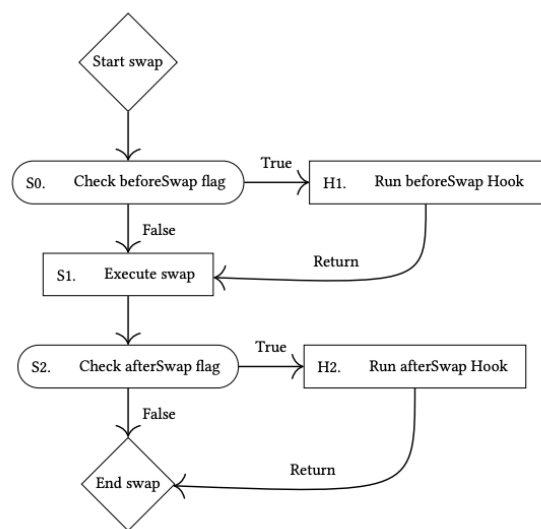


Figure 1: Swap Hook Flow

트랜잭션 데이터를 보면, 대부분의 사용자는 트랜잭션의 시작 또는 종료 시 ETH를 사용합니다.
이러한 상황은 불필요한 복잡성을 초래합니다.

5 사용자 정의 회계 (CUSTOM ACCOUNTING)

UNISWAP v4에서 새롭게 도입된 기능 중 하나는 사용자 정의 회계(Custom Accounting)입니다.
이 기능은 후크가 반환하는 델타(delta)를 이용해 사용자 동작을 변경할 수 있도록 후크 개발자에게 권한을 부여합니다.
델타는 사용자로부터 차감되고 후크에 적립되는 토큰 수량(또는 그 반대)으로 구성됩니다.
이를 통해 후크 개발자는 LP 포지션에 대한 출금 수수료, 맞춤형 LP 수수료 모델,
또는 특정 플로우(flow)와의 매칭 등 다양한 기능을 구현할 수 있으며,
결국 UNISWAP v4의 내장 집중형 유동성 모델을 그대로 활용할 수 있습니다.

중요하게도, 후크 개발자는 집중형 유동성 모델을 완전히 우회하여, v4 스왑 파라미터를 기반으로 사용자 정의 커브를 만들 수도 있습니다. 이러한 설계는 통합자에게 인터페이스 구성 가능성(interface composability)을 제공하며, 후크가 스왑 파라미터를 내부 로직에 매핑할 수 있도록 합니다.

UNISWAP v3에서는 사용자가 해당 버전에서 도입된 집중형 유동성 AMM을 반드시 사용해야 했습니다. 도입 이후, 집중형 유동성 AMM은 탈중앙화 금융 시장에서 기본적인 유동성 제공 전략으로 널리 채택되었습니다.

집중형 유동성은 대부분의 임의 유동성 전략을 지원할 수 있지만, 특정 전략을 구현하는 데 더 많은 가스 오버헤드가 발생할 수 있습니다.

가능한 예시 중 하나는 UNISWAP v2를 UNISWAP v4 후크 안에 구현하는 것으로, 내부 집중형 유동성 모델을 완전히 우회하여, 후크 내부에서 상수급 자동시장조성자(CPMM)를 구현하는 것입니다.

사용자 정의 회계를 사용하면, 동일한 전략을 집중형 유동성 수학으로 구현하는 것보다 더 비용 효율적입니다.

사용자 정의 AMM을 직접 구현하는 것과 비교할 때, 사용자 정의 회계를 사용하는 개발자에게 주어지는 이점은 *싱글톤(singleton)*, *플래시 회계(flash accounting)*,

그리고 *ERC-6909* 기능입니다.

이러한 기능은 더 저렴한 다중 홉 스왑, 보안성 향상, 그리고 플로우(flow)에 대한 더 쉬운 통합을 지원합니다.

또한 개발자는 자신이 구현할 AMM의 기반으로써, 보안 감사를 거친 코드베이스를 활용할 수 있습니다.

사용자 정의 회계는 유동성 제공 전략의 실험도 지원합니다.

기존에는 이러한 전략을 위해 완전히 새로운 AMM을 구축해야 했습니다.

그러나 사용자 정의 AMM을 만드는 것은 상당한 기술 자원과 비용 투자가 필요하며, 이는 많은 경우 경제적으로 현실적이지 않을 수 있습니다.

6 기타 주목할 만한 기능 (OTHER NOTABLE FEATURES)

6.1 ERC-6909 회계

UNISWAP v4는 ERC-6909 명세 [7]에 따라,

추가적인 토큰 회계를 위한 싱글톤 기반 ERC-6909 토큰의 발행(minting)과 소각(burning)을 지원합니다.

사용자는 이제 토큰을 싱글톤 내에 유지할 수 있으며,

컨트랙트와의 ERC-20 전송을 피할 수 있습니다.

이는 동일한 토큰을 여러 블록이나 트랜잭션에 걸쳐 반복적으로 사용하는 스왑 사용자, 유동성 공급자, 사용자 정의 회계 후크에 특히 유용할 것입니다.

6.2 거버넌스 업데이트

UNISWAP v3와 마찬가지로, UNISWAP v4에서도 거버넌스가 특정 풀에 대해 스왑 수수료의 일정 비율(상한 있음)을 가져갈 수 있도록 허용됩니다. 이 수수료는 LP 수수료에 추가되는 방식입니다. 단, UNISWAP v3와 달리, UNISWAP v4에서 거버넌스는 허용되는 수수료 계층(fee tier)이나 틱 간격(tick spacing)을 제어하지 않습니다.

6.3 가스 비용 절감

앞서 논의했듯이, UNISWAP v4는 플래시 회계, 싱글톤 모델, 네이티브 ETH 지원을 통해 의미 있는 가스 최적화를 달성합니다. 또한, 후크 도입으로 인해 UNISWAP v2 및 v3에 포함되어 있던 프로토콜 내장형 가격 오라클이 더 이상 필요하지 않게 되었습니다. 이에 따라, 기본 풀은 오라클을 완전히 생략할 수 있으며, 블록 내 해당 풀에서의 첫 스왑 시 약 15,000 가스가 절감됩니다.

6.4 donate()

donate() 함수는 사용자, 통합자, 후크가 풀의 두 토큰 중 하나 또는 모두를 사용하여 범위 내 유동성 공급자(in-range LP)에게 직접 보상을 지급할 수 있도록 합니다. 이 기능은 효율적인 지급을 위한 수수료 회계 시스템에 기반합니다. 해당 수수료 지급 시스템은 풀의 토큰 페어 중 하나만을 지원할 수 있습니다. 가능한 사용 사례로는, TWAMM 주문에서 범위 내 LP에게 팁을 주거나, 새로운 형태의 수수료 시스템을 구현하는 것 등이 있습니다.

7 요약 (SUMMARY)

요약하자면, UNISWAP v4는 비(非)커스터디형, 업그레이드 불가능, 퍼미션리스(permissionless) AMM 프로토콜입니다. UNISWAP v3에서 도입된 집중형 유동성 모델을 기반으로, 후크를 통해 커스터마이징 가능한 풀을 제공합니다. 후크를 보완하는 구조적 변경 사항으로는, 모든 풀 상태를 하나의 컨트랙트 내에서 관리하는 싱글톤(singleton)과, 각 풀의 지급 능력을 효율적으로 보장하는 플래시 회계(flash accounting)가 있습니다. 또한, 후크 개발자는 집중형 유동성 모델을 완전히 우회하고, v4 싱글톤을 임의 델타 해결기(delta resolver)로 활용할 수 있습니다. 기타 주요 개선 사항으로는 네이티브 ETH 지원, ERC-6909 기반 잔액 회계, 새로운 수수료 메커니즘, 그리고 범위 내 유동성 공급자에게 기부할 수 있는 기능 등이 포함됩니다.

참고 문헌 (REFERENCES)

- [1] Austin Adams, Ciamac Moallemi, Sara Reynolds, and Dan Robinson. 2024. am-AMM: 경매 기반 자동화 시장조성자(Auction-Managed Automated Market Maker). *arXiv preprint arXiv:2403.03367* (2024).
- [2] Hayden Adams. 2018. *Uniswap v1 Core*. 2023년 6월 12일 확인.
<https://hackmd.io/@HaydenAdams/HJ9jLsfTz>
- [3] Hayden Adams, Noah Zinsmeister, and Dan Robinson. 2020. *Uniswap v2 Core*. 2023년 6월 12일 확인.
<https://uniswap.org/whitepaper.pdf>
- [4] Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson. 2021. *Uniswap v3 Core*. 2023년 6월 12일 확인.
<https://uniswap.org/whitepaper-v3.pdf>
- [5] Alexey Akhunov and Moody Salem. 2018. *EIP-1153: 일시적 저장 연산자(Transient storage opcodes)*. 2023년 6월 12일 확인.
<https://eips.ethereum.org/EIPS/eip-1153>
- [6] Vitalik Buterin and Martin Swende. 2021. *EIP-3529: 저장소 환불 감소(Reduction in refunds)*. 2023년 6월 12일 확인.
<https://eips.ethereum.org/EIPS/eip-3529>
- [7] JT Riley, Dillon, Sara, Vectorized, and Neodaoist. 2023. *ERC-6909: 최소 다중 토큰 인터페이스(Minimal Multi-Token Interface)*. 2024년 8월 26일 확인.
<https://eips.ethereum.org/EIPS/eip-6909>
- [8] Dave White, Dan Robinson, and Hayden Adams. 2021. *TWAMM*. 2023년 6월 12일 확인.
<https://www.paradigm.xyz/2021/07/twamm>

면책 조항 (DISCLAIMER)

본 문서는 일반 정보 제공 목적일 뿐입니다.
이는 투자 조언, 권고, 또는 어떤 투자에 대한 매수/매도 권유가 아니며,
어떠한 투자 결정을 평가하는 데 사용되어서는 안 됩니다.
회계, 법률, 세무 자문 또는 투자 권고를 위한 목적으로 사용되어서는 안 됩니다.
본 문서는 작성자의 현재 의견을 반영하며,
Uniswap Labs, Paradigm 또는 이들의 계열사를 대표하지 않으며,
그들과 관련된 개인 또는 조직의 의견을 반드시 반영하지는 않습니다.
본 문서에 반영된 의견은 예고 없이 변경될 수 있습니다.