Manifest: The Orderbook Manifesto

CKS Systems
cks@cks.systems
www.manifest.trade

Abstract: Manifest implements an optimized feature set for an on-chain orderbook exchange by maximizing capital efficiency through atomic just-in-time (JIT) liquidity. We introduce the Hypertree, a data structure which enables orders of magnitude more efficient key-value data storage on the Solana Virtual Machine (SVM). A layered architecture allows the core matching engine to be formally verified without compromising on features. Manifest is open source and a free public good. Zero protocol fees guarantee that all market participants have equal access and opportunity. Cheap and permissionless market creation enables new orderbook use cases. The program deployment to Solana accelerates the transition for all risk-transfer to occur efficiently and securely on-chain.

1. Introduction

Manifest is the 3rd-generation on-chain orderbook built on Solana. An orderbook represents the purest form of risk-transfer and is the most suitable market design for a base liquidity primitive. If an orderbook can be fast, cheap and efficient it offers more desirable properties for trading activity than automated market makers. By existing on-chain, an orderbook inherits non-custodial, permissionless and sequential processing features, unlocking trustless and transparent markets for all participants equally. Global orders enable traders to provide liquidity to as many markets as they desire without being burdened by capital constraints.

Market creation cost and complexity has been simplified through reallocation and the Hypertree data structure, as well as a novel matching algorithm that doesn't require pre-configured tick and lot sizes.

To enable the vast feature set that is required from modern decentralized exchanges without compromising security or performance, a layered architecture was implemented.

1.1 Background

In 2024, there exist two network variants and two risk-transfer designs for crypto trading applications and protocols.

Network variants are either off-chain or on-chain, also sometimes called centralized or decentralized. Exchanges exist on a spectrum of centralization regardless of the network they operate on being run by a set of decentralized nodes. Decentralization is non-binary. Off-chain refers to a non-public ledger or company database for trade matching and on-chain refers to a fully auditable public ledger. The tradeoffs historically have been performance-based for why an exchange chooses to be off-chain, rather than on-chain. Additionally by being on-chain, traders give up absolute anonymity and in return your counterparty is always known.

Risk-transfer designs fall into two main categories, Automated Market Makers (AMMs) and Central Limit Order Books (CLOBs), referred to interchangeably as simply “orderbooks”. AMMs are an innovation within crypto, popularized by Uniswap on Ethereum. The main rationale for the existence of AMMs is the limitation of a blockchain network to support high throughput order matching. This workaround has led to many interesting applications, but ultimately does not afford users the ability to express risk as clearly and along as many dimensions as an orderbook. The AMM design is simple and inclusive, but lacks precision and capital efficiency.

Exchanges	AMM	Orderbook
Off-Chain	Not Useful. Not Built	Binance, Coinbase, Kraken, Mt.Gox, FTX
On-Chain	Uniswap, Sushi, Raydium, Orca	EtherDelta, Openbook, Phoenix, Manifest

Figure 1: Exchange Variants. Network by Risk-Transfer

1.2 Generations of Orderbooks

EtherDelta was the origin and 1st-generation for on-chain orderbooks. Built on Ethereum, however, the exchange was notoriously difficult to use due to the reliance on off-chain matching and even the on-chain settlement did not scale due to Ethereum's throughput and transaction costs. Nonetheless, they proved the concept and identified clear improvements for following projects and networks to tackle.

Solana enabled the ability to cheaply and quickly process transactions and thus gave birth to the first high performance orderbook in the form of Serum. This 2nd-generation orderbook on Solana was a

stepwise improvement for on-chain risk-transfer and showcased what could be done on Solana, but still required makers to settle their trades after matching, through a second transaction. A few days after the FTX collapse, Serum was forked and relaunched as Openbook by a cohort of Solana DeFi contributors due to an uncertain upgrade authority - an ever present risk for programs that are not made immutable.

The design was significantly improved in subsequent iterations, namely Mango, Phoenix and OpenbookV2. These protocols also addressed major pain points of Serum like the lack of time-in-force orders, cranking cost and asynchronous settlement. However, they were overburdened with features and costs that could be stripped out for a more streamlined risk matching program.

Manifest is the 3rd-generation orderbook and the start of the endgame for on-chain risk-transfer. The key design innovations being the increased capital efficiency allows unlimited rehypothecation without any oracle dependency. At the same time, it improves the expressible price range through the introduction of subatomic ticks which makes it more competitive in providing liquidity to searchers and on-chain arbitrage traders.

The orderbook implementation is at the theoretical limits of Solana in several dimensions and approaching it in others. These include number of accounts required, instruction call data size, rent, trading fees, trade sizes, price ticks and composability.

1.3 Hybrids

Two alternative solutions to the throughput issues of fully on-chain orderbooks exist and are worth highlighting in contrast to Manifest. First is simply to make an L2 or appchain. Second is to separate the matching and settlement of the trades. These have become the dominant design choice in other ecosystems with the likes of DyDx, GMX, HyperLiquid, Aori etc. with all of them at some point implementing some flavor of these shortcuts. Even Uniswap has ventured into the hybrid on-chain/off-chain space with UniswapX, its off-chain request-for-quote (RFQ) style matching system.

On one hand, these designs are pragmatic; on the other hand, they jeopardize the primary rationale for building on-chain financial infrastructure - transparency and trustlessness. UniswapX successfully avoids gas fees, but those gas fees just get repackaged as a spread charged on the orders by liquidity providers. Traders should always be able to see their counterparty and understand at what effective price the other side is willing to trade. By introducing barriers to the full order lifecycle, these exchanges fragment liquidity and prevent a more capital and price efficient market from developing.

1.4 Motivation

First-generation orderbooks were designed for a world that was totally different from our current and future on-chain reality. Historically, on Solana, gas and rent cost was cheap, but in many regards that has recently changed. From that lens, Manifest - designed with a ~500x reduction in market creation rent cost, ~50% reduction in gas, compute, and infinite capital efficiency gain does not seem revolutionary, but rather an overdue necessity to build.

The design specifications became clear after extensively trading on OpenbookV2 and Phoenix and determining a significantly optimized orderbook was possible only through complete program redesign. We named this endgame exchange Manifest for the promise it holds to cement the superiority of on-chain trading. The goal is to bring all liquidity and all order flow on-chain by building an unrivaled orderbook primitive as an open-source public good. We wanted to push the theoretical boundaries of what is possible on Solana and determine if its initial vision of NASDAQ at the speed of light is still feasible.

1.5 Opportunity

With Manifest, the playing field is tilted in favor of on-chain versus off-chain exchanges for the first time in history. In the worst case, Manifest should capture similar volumes to Openbook, about ~15% of daily volume on Solana. With no barriers to list and no protocol fees, the implemented program design ensures there is absolutely no reason for every SPL token not to have a Manifest market.

Manifest harbors much bolder potential as the dominant alternative to centralized exchanges, AMMs, and hybrid DEXs. Given its feeless, capital efficient and compute optimized design, Manifest has a distinct possibility to become Solana's base liquidity primitive of the future and fulfill a destiny to reconstitute all crypto liquidity on-chain.

2. Design Goals

Manifest development begins by determining the foundation on which features need to be implemented.

2.1 Layered Architecture

The design of Manifest is centered around keeping bloat away from the core program. It accomplishes this by utilizing a wrapper program that handles more complex features and composes on the stripped-down core. Any instructions that can be handled at higher layers on the stack will not be executed in Manifest's core. Thus, Manifest implements the orderbook as a base liquidity primitive, rather than on the application layer.

By virtue of this modular design, Manifest supports traders who require specific features without burdening the security and performance of order matching. Traders should interact with Manifest on-chain using a wrapper program. Features like ClientOrderId, FillOrKill, PostOnlySlide, adjusting orders for insufficient funds can and should be in a wrapper program that does a Cross-Program Invocation (CPI) into the core. Manifest does not have silent failures in the core, it fails on all failures, but still allows those who want to manage them in a wrapper to do so.

2.2 Zero Protocol Fees

Manifest does not impose any protocol fees for traders within the core program. Both makers and takers have free access to transact with each other at all times. This design choice was made to provide a

forever free alternative to existing crypto exchanges. Makers and takers alike do not have to factor in fees into their quoted spreads or tolerable slippages, making the user experience simpler.

A wrapper program is fully customizable and could impose fees if its developers choose. However, the reference wrapper implementation is feeless and will always remain that way for users to directly access Manifest without fees.

2.3 Expandable Accounts

Market creation fees, or the rent and gas required to create a new market, are minimized on Manifest. Compared to 2nd-generation orderbooks, Manifest provides a ~1000x improvement on market creation fees, costing only about 0.004 SOL in rent to initialize a new market. This is because Phoenix pre-allocates the maximum number of orders on the bids and asks booksides at creation time. OpenbookV2 has that as well as an event queue.

This feature is enabled through the use of realloc on Solana, which allows for expandable accounts. No longer do markets need to have large accounts created at initialization with high rent. Instead, the rent can be incrementally added to expand these market accounts as more space is needed for more orders and seats for traders.

In the case of an extremely active market, this design could still add up to a meaningful amount of rent accumulated, but that rent was paid incrementally as needed by traders, not the creator. Existing popular markets on 2nd-generation orderbooks often have less than 100 orders on them, so in the steady state we do not expect the aggregate rent for a Manifest market to be as much still.

Figure 2: Number of markets holding 100 orders that can be created with 12 SOL

As a result of the low market creation cost, Manifest opens up orderbooks to an entirely new set of tokens for which it previously did not make economic sense to create a market. In fact, an application like pump.fun which partially exists to reduce wasted capital from creating a Raydium pool, if the token will never become popular, now provides an extremely cheap route for every token to have an orderbook. New low-liquidity use cases we do not foresee yet will arise, along with some obvious ones like lending, options, liquid staking, NFTs, memecoins etc. Every SPL token can now be cheaply, trustlessly and permissionlessly offered for purchase or bid on unsolicited.

2.4 Global Orders

Global orders allow the same token to be used to provide a resting order on an unlimited number of markets as depicted in Figure 1. This is enabled by being on-chain and having sequential processing such that these orders from a shared deposit account cannot match more than once. Traders on Manifest effectively have infinite capital efficiency and initial margin to trade across markets.

For example, a market maker can now utilize the same bucket of USDC to provide a bid for SOL/USDC, BONK/USDC, and any number of other markets. Global orders apply to both bids and asks, so SOL can be reused to show a bonkSOL/SOL bid and an ask on SOL/USDC.

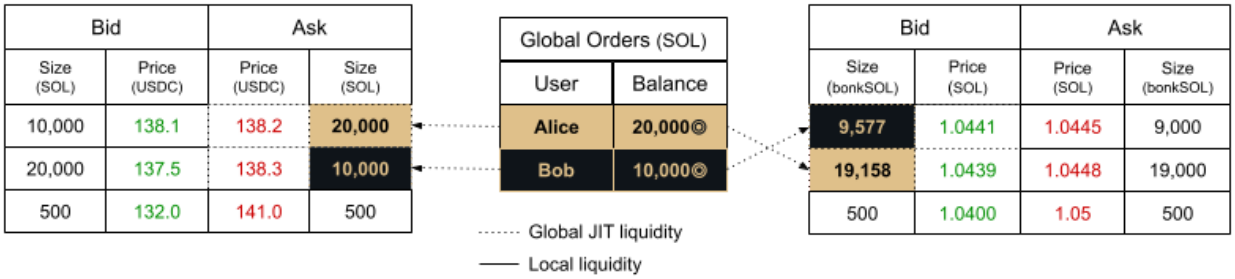

Figure 3: Global Orders providing JIT liquidity in multiple markets

For growing stablecoins (USDS, PYUSD, etc.) or yieldcoins (CHAI, USDY, etc.) global orders will help their quote liquidity proliferate since less capital is required, and it can be allocated to bidding on unlimited direct pairs. No longer will holding and trading these tokens require a swap with USDC to access most liquidity.

For liquid staking tokens (LSTs), global orders may have the most outsized liquidity benefit. SOL bid liquidity can now be shared infinitely across different LSTs. The result is that every LSTs could be bid on without locking up inordinate amounts of capital.

In effect, global orders are a form of just-in-time (JIT) liquidity. They only require traders to deposit their funds to the global order account, and from there, they can be used on any combination of markets. The program prevents any double spend or duplicate matching, but does leave the possibility for unfillable orders to remain on the book. In response to this and to reduce spam, global orders prepay their gas cost for their removal. The program will remove unfillable orders at the time of matching, as well as allow other users to claim this extra gas by cleaning up stale orders any time.

Total value locked also becomes a less meaningful measurement of protocol value on Manifest because of global orders. This capital efficiency also increases the cost of attacking the protocol by having less funds at risk and reduces the cost for a market maker to deploy strategies on Manifest, especially on lower liquidity tokens. Capital efficiency is unlimited as it grows with the number of markets in existence, which from section 2.3 we've shown should be extremely large.

Global orders present a new paradigm for composability and capital efficiency. As the trend of intents and hybrid exchanges moves more solutions off-chain reducing their composability and shared

liquidity, Manifest is moving DeFi back in the other direction towards a unified and capital efficient liquidity primitive.

2.5 Order Precision

A key insight from experience on 2nd-generation Solana orderbooks is that order routing from aggregators like Jupiter and on-chain arbitrage traders is sensitive to certain parameters. In particular, because orderbooks are discrete in price and size, as opposed to continuous for an AMM, smaller tick and minimum order sizes integrate better for receiving order flow. All prior implementations were in practice limited by a tradeoff: tick size and minimum order size are inversely proportional in order to avoid subatomic trade outcomes.

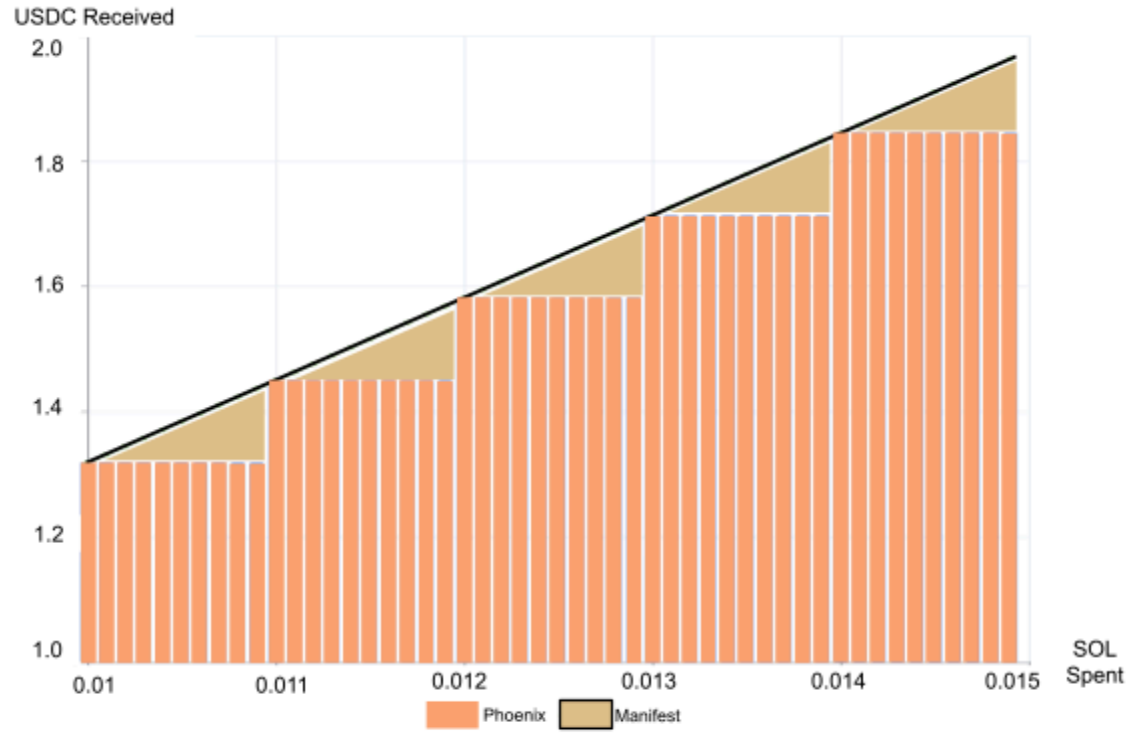

Figure 4: Swap Input vs Output Tick Design Comparison

Manifest removes the tradeoff between large ticks or large minimum order quantities altogether by making minimum order size atomic and tick size subatomic.

$$Price\ Tick\ Size\ = 10^{-18}\ \text{Quote Atoms / Base Atom}$$
$$Min\ Order\ Size\ =\ 1$$

Trade outcomes are rounded towards the maker to protect liquidity providers as they are not in control of the matching order size. Internally the order prices are represented as 128 bit fixed point numbers with 18 fractional decimals. Which allows competitive precision to AMMs that often use a 128 bit binary fixed point format with 64 fractional bits.

Removal of independently determined tick size and min order quantity has the convenient feature of reducing complexity for market creation. In order to create a market for a particular token pair, you only need to input the token mints.

3. Implementation

Manifest is a complete redesign of orderbooks on Solana informed by improvements made in prior generations. The program introduces a new data structure called a Hypertree. The program code is open source under GPL.

3.1 Architecture

The program features a modular design with a core + wrapper program. The core program contains the matching engine and basic instructions and is protected and segregated from a wrapper that allows unlimited diversity of feature sets for any trader or interface. Only absolutely necessary features were included in the core program allowing for formal verification.

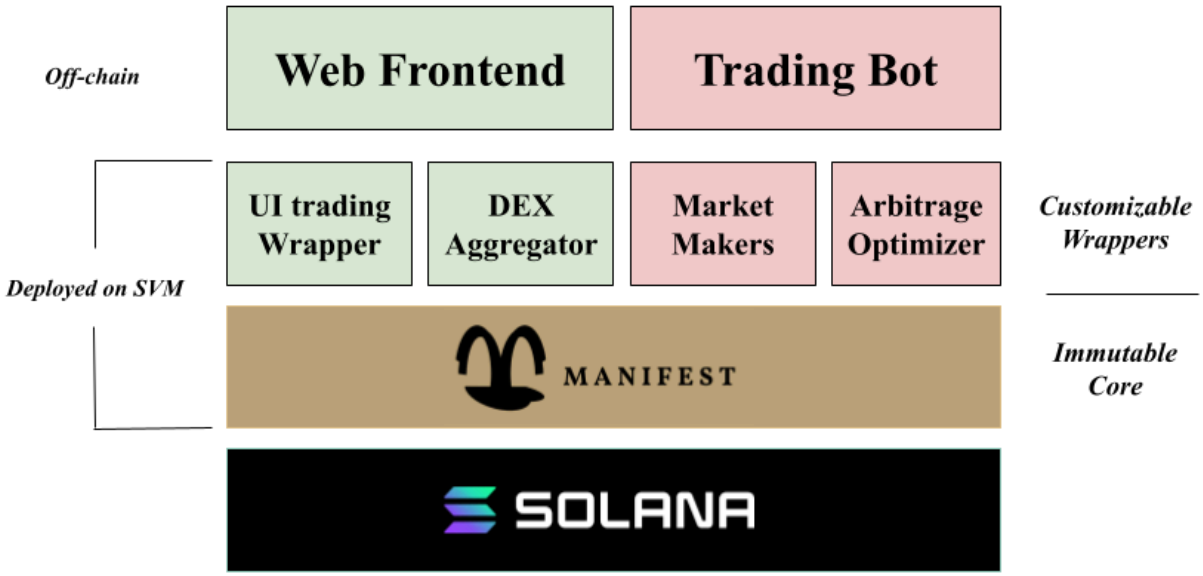

Figure 5: Technical Architecture, Core + Wrapper

The formal verification of the core program is in progress. Once complete it can be considered maximally secure for wrappers to compose on. A reference implementation and deployment of a wrapper are provided, guiding third party developers in developing their own on-chain components for their desired target application. The reference wrapper implements ClientOrderId cancellation and automatic adjustment of orders for insufficient funds which are useful for market makers.

The use cases of the reference wrapper can be extended to UIs that want to enable special features

like volume or token ownership based fees, or arbitrage traders who want to optimize their trade size to the liquidity available on a market.

3.2 Matching Engine

Orders that take liquidity from existing resting orders are matched in the core program. Base atoms are always exactly matched. Due to atomic order sizes and subatomic prices, rounding quote amounts to fill orders may be required. Any required rounding on partial fills is in favor of makers and full fills in favor of takers. The rounding is bounded to 1 quote atom per matched order pair. When bids receive rounding in their favor, the quote quantity paid is rounded down. The reverse takes place on the ask side: the quote quantity paid by is rounded up.

Thus, we can imagine a situation where multiple better priced maker orders should not get matched because a single larger order should avoid cumulative rounding. One possible solution is to optimize each taker order across the whole book, but it would increase the runtime complexity too much. Instead the orderbook is not strictly sorted by price-time priority, but uses the effective price of each order for ranking.

$$Effective\ Price\ =\ floor(\ round_{m}(\ price\ x\ base\_atoms\ )\ /\ base\_atoms\ )$$

To illustrate, suppose Manifest has the following bids, the left table is sorted by the price, while the right table is sorted by the effective price like the orderbook of Manifest. All prices are in base atoms / quote atoms.

Bids		
Size (Atoms)	Price	Effective Price
1	1.200	1.0000
10	1.191	1.1000
20	1.190	1.1500
30	1.180	1.1667

Effective Bids		
Size (Atoms)	Price	Effective Price
30	1.180	1.1667
20	1.190	1.1500
10	1.191	1.1000
1	1.200	1.0000

The effective price sorting, changes the quote amount the taker would receive. Matching a taker sell order of 30 atoms would yield 34 in price order, but 35 in effective price order. In practice this will not impact most trades, due to normal order sizes being significantly above 10^6 quote atoms. It prevents exploiting the rounding through placement of small orders or small price increments alike.

To verify the integrity of this change, randomized tests with the following invariant were implemented and run for multiple weeks inside a minimal SVM implementation:

I: *For any random series of PlaceOrder, CancelOrder, Swap verify that if all remaining orders were canceled and every user would withdraw all remaining balances, the sum of all balances is the same as before executing the random series.*

II: *Given an order of random price (price = a x 10^b, a:u32, b:[-18,8]) and size in base atoms (u64) placed on any side of the book, verify for any random series of orders placed on the same side, that a taker order would be able to trade at the given or a better price adjusted by one quote atom for each resting order with a better effective price.*

3.3 Data Structure

The innovation that allows Manifest's leap in on-chain trading is the Hypertree. All data in the market account fits into graph nodes of the same size (80 bytes), which lets independent data structures grow without being fully initialized from the start by interleaving.

The Hypertree is a library for creating efficient data structures on Solana that do not own the underlying byte array. This special feature allows overlapping data structures within the same region of memory, inside of the same account. A Hypertree does not require contiguous memory and implements max, insert, delete, lookup, iterator operators. Read and write operations can be separated.

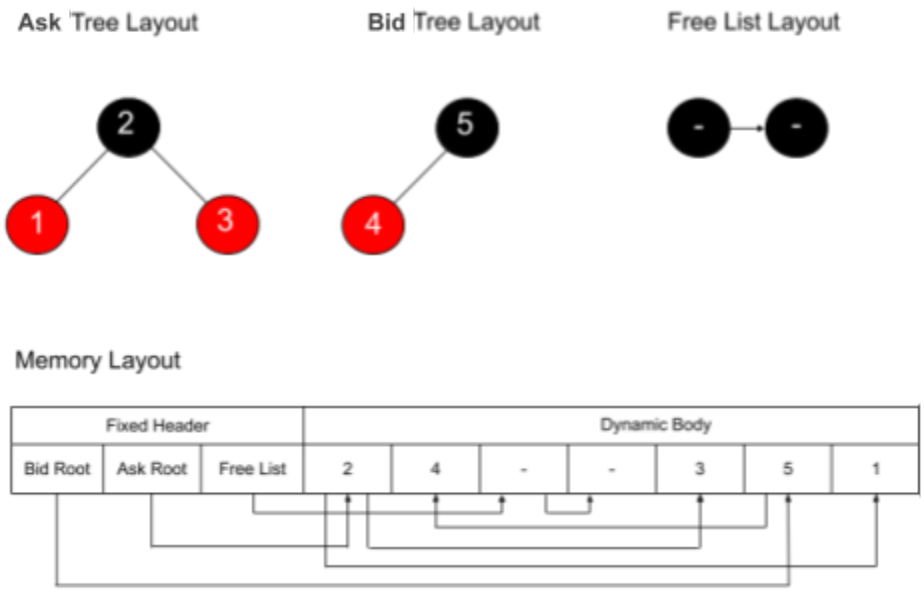

Figure 6: Hypertree Layout

The market account holds all relevant information. It begins with a header that stores all of the fixed information for the market, like *BaseMint*, *QuoteMint*. All variable data, *RestingOrders* and *ClaimedSeats*, are in the dynamic byte array after the header. These are three Red-Black trees for *Bids*, *Asks*, and *ClaimedSeats* and one Linked-List for *FreeListNodes*, overlapping across each other. All are

graphs where each vertex, along with an adjacency list, fits in 80 bytes, allowing them to use the same blocks.

3.4 Safeguards

Usage patterns on Manifest will be different than prior generations of on-chain orderbooks since fees are so drastically reduced. Without the rent burden, anyone can create a market, which although not bad on its own, could lead to fragmentation and spam. Placing orders becomes cheaper and because of maker rounding it could be profitable to spam tiny orders. Global orders have minimal cost but need to ensure they are used to provide viable liquidity. As a result, costs using prior orderbooks were not all bad, they incentivized user behavior into purely valuable economic activity. However, due to Manifest's efficiency and low cost, we introduce these new problems to address. While higher performance off-chain exchanges can just rate-limit these spam attacks, Manifest must correct for these behaviors and avoid these potential attacks internally.

Rent and gas prepayment are used as the economic disincentive for security against denial of service activities. Market griefing spam is deterred by the unrecoverable rent required to expand markets. Non-viable order spam can be a nuisance, but if a market were ever full, good actors can simply make a new market and move orders there. The spammer then loses their gas and rent from spam that grew the spammed market's account size.

Global order spam is deterred by gas prepayment, such that if you do not clean up your orders when they are not able to be filled, another user or matched order can claim the gas you prepaid.

Global order seat eviction is deterred by requiring a deposit more than the minimum existing deposit to claim a seat. If the seats are ever full, whoever has the least deposited gets evicted by the new user. This means that, in order to deny use of the global order account by evicting everyone you would require absurd amounts of capital cumulatively and too many orders to fit into one transaction for flash loans to be useful.

To identify and combat wash trading on feeless trading, thorough indexing and pattern detection can be implemented up the stack. Manifest's absolute transparency of having the full trade lifecycle on-chain will make it easier to flag manipulative trading behaviors than centralized alternatives.

3.5 Network Requirements

Solana requires all accounts to be known before creating a transaction. This design constraint can be circumvented by placing all data into a Hypertree within a Solana account. Using reallocation, rent for market accounts can be minimized at creation. Each time a new trader requires a seat or more space for orders, they contribute to the growing size by paying its incremental rent.

Manifest is written in a compute unit optimized manner without the use of Anchor. The best practice for advanced programs on Solana is no longer to take the compute tradeoff for the convenience of using Anchor.

4. Features

Key improvements outside of the design goals in Section 2, were informed by extensive trading on prior generations of orderbooks. The feature set implemented accomplishes the goal of an orderbook protocol that cannot be significantly improved and whose codebase is worthy of being enshrined and made immutable.

4.1 Minimal Transaction Size

The main parameter to maximize order routing from aggregators is to minimize the accounts required in swap instructions. This is due to Solana's 32 account maximum in a single transaction.

The other parameter is instruction call data which was minimized through the usage of a custom floating decimal point data type, an unsigned 32 bit mantissa with a signed 8 bit exponent. This means for routers that enable multiple hops, exchanges with more accounts or larger call data are more likely to breach this limit and therefore less likely to receive order flow. Even if there's a better price, it can still get penalized in these multiple hop route situations.

Manifest mitigates this integration load by pushing the required accounts to the theoretical limit for an orderbook. Manifest uses 7 accounts to route, when not involving a global order or Token2022 market, 1 fewer than Phoenix and 9 fewer than OpenbookV2.

4.2 Compute Optimized

Compute units (CU) spent per order is a critical metric to minimize for orderbook transactions. Market makers are the most active users and largest consumers of CU due to frequently canceling and replacing orders. These instructions account for more than 95% of total compute spent on orderbooks. Order fills however, are the most economically valuable and most expensive operation. Fills require iterating, rebalancing and reinserting. Makers need to budget their transactions before sending to avoid underestimation, thus they care about higher percentile CU usage.

Optimizations across all instructions were implemented, like index hints to avoid lookups which speeds up the Hypertree to constant time, O(1). The result for high CU spenders is that their worst case CU usage has been significantly improved while still allowing for maximal order matching and fills.

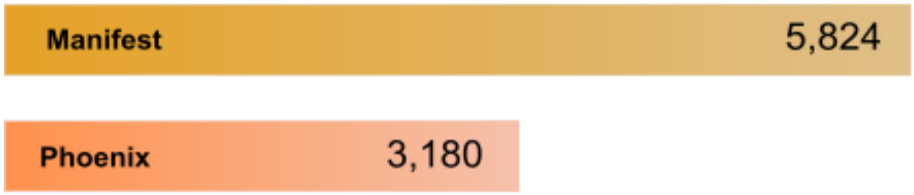

Figure 7: Estimated capacity for orders per block based on 95th percentile compute cost

Manifest continues to collect benchmarking data against Phoenix which is linked in section 7.2. Recent benchmarks against past real world trading data show Manifest using 45% less CU per order than Phoenix. We only benchmark against Phoenix because it is strictly better than OpenbookV2 on CU usage.

4.3 Token Extensions

Despite being a point of contention in DeFi, the Token2022 standard is an important Solana feature to support given the recent growth of PYUSD and to future-proof the exchange. Manifest supports token extensions without compromising the efficiency gains elsewhere in the program when not using it. Incorporating future token programs into Manifest is made easier by this support, such as the highly anticipated nano-token program.

4.4 Crankless

The crankless features pioneered by Ellipsis Labs for the aforementioned Phoenix orderbook, continue on Manifest. All users are included in one account using the Hypertree data structure. The key benefit here is no off-chain services are required for the orderbook to function, enabling instant atomic settlement.

4.5 Read Optimized

The only remaining rationale for not being crankless was that open orders account separation is a necessary feature for margin trading. Read locks on accounts to get the open orders for a trader are frequent on a margin exchange. The default wrapper implementation of Manifest allows a margin exchange to check on orders and balances without read locking the market for everyone. This makes a crankless design now strictly superior.

5. Maintenance

Manifest is designed to require no ongoing maintenance or off-chain services for the core program and reference wrapper implementation. Additional wrappers should be publicly maintained by teams actively using the exchange.

5.1 Tokenomics

Manifest is a public good, open-source blockchain protocol whose development was not externally funded, is not seeking investment and will not incur future costs. This economic purity is important. Manifest represents DeFi's best chance at disrupting centralized exchange trading both in crypto and traditional markets. We believe a protocol token risks corrupting that.

Fortunately, Manifest does not require a token for governance or as an economic incentive within the protocol. The program without a token is even more resilient and anti-fragile. Therefore, Manifest as an immutable orderbook protocol will not have a token.

Tokens are more suitable for evangelizing a community and building culture around an idea or application. Manifest is supportive of this by virtue of being permissionless and cheap. The protocol is agnostic to listing tokens, nor does it preclude the client layer wrappers or frontends from integrating one.

5.2 Immutability

In the short term, governance or access to upgrade the core program will be maintained by the development team. After formal verification, the goal is to make the core program immutable. Locking the program from having future changes comes with risk of non-withdrawable funds that upgradable programs avoid. However, the benefits of immutability, namely no maintenance and no governance, far outweigh the costs which are significantly mitigated through formal verification. The expandable account design is a mitigant that ensures it is trivial and cheap to switch to new markets, if ever needed. Oracles are never a dependency for the protocol. The formal verification of the core, extensive fuzzing and testing will give us confidence that a loss of funds and inaccurate matching is not possible and result in making Manifest immutable.

5.3 Status

The program is being deployed to mainnet soon after the release of this whitepaper. Manifest core program address is MNFSTqtC93rEfYHB6hF82sKdZpUDFWkViLByLd1k1Ms. The reference wrapper program address is wMNFSTkir3HgyZTsB7uqu3i7FA73grFCptPXgrZjksL. The deployer address and upgrade authority is B6dmr2UAn2wgjdm3T4N1Vjd8oPYRRTguByW7AEngkeL6.

The program is currently under audit for formal verification by Certora with an expected completion November 1st, 2024. They have previously formally verified programs on Solana, including Token2022 and SquadsV3.

5.4 Contributions

Development was led pro-bono by CKS Systems to provide a better trading experience for projects and their users that want liquid on-chain orderbooks. We welcome contributions of all shapes and sizes to continue advancing and promoting this orderbook technology and the benefits it holds for the Solana ecosystem.

There is no way to invest in Manifest besides contributing your time and energy to it. Partners to build with are welcome.

6. Conclusion

Manifest provides an endgame orderbook liquidity primitive that accelerates the transition of all risk-transfer to occur on-chain.

Specifications	OpenbookV2	Phoenix	Manifest
Feeless	No	No	Yes*
Market Rent	2 SOL	~3 SOL	0.004 SOL*
Capital Efficiency	Single Orders	Single Orders	Global Orders*
Composable Wrapper	No	No	Yes
Silent Failures	Allowed	Allowed	Disabled
Tick Size	Required	Required	Subatomic*
Minimum Quantity	Required	Required	Atomic*
Swap Accounts	16	8	7*
Compute Optimized	Low	High	Very High*
Token 22	Unsupported	Unsupported	Supported
Crankless	No	Yes	Yes
Read Optimized	Yes	No	Yes
Anchor	Yes	No	No
License	GPL	Business	GPL

Figure 8: Comparison of Key Features. * denotes value is at or near theoretical limits.

6.1 Mission

We set out to deliver a set of features for an orderbook exchange on Solana that will never need to be improved and can be confidently made immutable. We accomplished this by developing a Hypertree data structure, a secure core + wrapper architecture, and a compute and account optimized approach to Solana. The exchange implementation resulted in an open-source, feeless, cheap market creation, capital efficient orderbook primitive.

Our mission is larger than just the technical robustness of this on-chain exchange. We hope to inspire builders to compose innovative new applications on Manifest that have now been unlocked by its features. On a broader scale, we believe this spirit of open innovation has the potential to disrupt the crypto cartel, those who are not aligned with the ethos of crypto and run permissioned, rent extractive, off-chain products and services.

6.2 Manifest Together

This document is called a Manifesto, not just for the play on words, but an honest philosophical belief in the good that programs like Manifest represent to everyone's transactional well-being. Built at a time in crypto's history when narratives have shifted from "crypto is going to save the world" to "crypto is a tool for speculation wrought with financial nihilism," Manifest hopes to be a beacon for degens and nerds alike to agree on what matters in this industry: innovative, permissionless, trustless, value-additive systems.

Manifest proves that there still is much to be built in crypto and there still are people eager to build these technologies simply for the betterment of on-chain experiences and the public good. In the meantime, each and everyone of us who believes in the libertarian principles of crypto can make a difference by taking custody and power away from centralized crypto & financial companies by moving trading activity on-chain. Let centralized exchanges be but a footnote in the history of the meteoric rise of crypto, serving merely as fiat on- and off-ramps to access the greatest liquidity layer the world has ever built on Manifest.

7. Appendix

7.1 FAQ

What is Manifest?

Short answer: An orderbook (CLOB DEX) built on Solana. Long answer: See Above

Why should I care about Manifest?

Never before has an on-chain exchange (DEX) been as cheap, fast and efficient as Manifest. Trade with zero fees, create new markets for next to nothing, and use the same tokens to place orders on different markets. On-chain trading using Manifest is now a viable threat to the existence of centralized exchanges.

How can I use Manifest?

Swap as you normally do through Jupiter or Autobahn (upcoming) and automatically route to Manifest. Create new markets for your tokens and place actual limit orders on-chain via the SDK, separately hosted UIs such as Fill City or run your own.

Who created Manifest?

CKS Systems, a blockchain native market maker developed the Manifest program to have a better on-chain orderbook to trade and market make on.

When is Manifest launching?

The program is live on Solana devnet and launching on mainnet soon after the publication of this whitepaper, likely at Breakpoint 2024. The program is expected to be formally audited by the end of October 2024 and made immutable thereafter.

How is Manifest different from AMMs?

Manifest is an orderbook (CLOB) which allows traders to place orders to buy and sell at specific prices and for specific amounts. These are called limit orders and are the same style of trading commonplace on any centralized exchange. Unlike AMMs, orderbooks do not immediately convert a buy to a sell order or vice versa when it gets filled. Orderbooks are preferred by sophisticated traders because they allow more exact risk management.

What is formal verification and why is it important for Manifest?

Formal verification means that the orderbook's code has been mathematically proven to work as intended, without bugs or vulnerabilities. It is important because it ensures Manifest can be made immutable, meaning it can run securely on-chain without needing updates, changes or any governance, providing long-term reliability and trust. Certora is independently working on completing this.

What is the main technical innovation that powers Manifest?

The Hypertree data structure organizes market data in a way that allows it to grow and adapt as needed, ensuring smooth and efficient trading, no matter how large the market becomes.

What is the main financial innovation that powers Manifest?

Global orders unleash unparalleled capital efficiency for a spot exchange. Reusing the same funds to provide liquidity across markets means the cost of trading on-chain for liquidity providers is minimized.

What is the main security innovation that powers Manifest?

Modular program design that offloads complex features into a client layer. Manifest's core is infrastructure level code and wrappers are the client layer where applications are built.

How can I help Manifest?

Trade on-chain. Self-custody your funds. Don't pay exchange fees anymore. Spread the news that crypto is alive and well today, building and solving important problems to make a better, more equitable financial future for everyone.

How can I get liquidity for my token on Manifest?

Tell your community to place limit orders using Manifest. Work with any experienced **on-chain** market maker. Contract one entirely on-chain with Dual Finance.

Where can I find out more about Manifest?

Visit manifest.trade, follow x.com/ManifestTrade, star github.com/CKS-Systems/manifest

How can I get in touch with the team?

Fill out this inquiry form.

7.2 Links

Website: manifest.trade

CU Benchmarks: https://cks-systems.github.io/manifest/dev/bench/

Press Release: https://cks.systems/Manifest_Launch-Press_Release.pdf

8. Revisions

8.1 October 20, 2024

Effective Price, Section 3.2, methodology was discontinued following auditor suggestions. A new rounding scheme was introduced which drastically simplifies. All rounding of excess atoms is done in favor of the maker, except when fully filling a resting order, then it is in favor of the taker. In this setup, if a maker has an order for an even number of atoms, they will always get at least that since there cannot be just rounding against them. This allows sorting to be done strictly on the limit price, avoiding the complication of re-sorting the tree on partial fills.

Silent failures are introduced in limited cases. Because acquiring write locks for global accounts may be expensive, takers may choose not to include them. In this case, the taker will be allowed to fill through the orderbook until it needs to interact with a global order and then terminate, instead of reverting. This change is done because it is a type of error that a wrapper program cannot as easily prevent. Also, the program design is to be as attractive as possible to router flow, and that does not use wrappers.