Introduction

The current smart wallet experience for users is not suited for transacting in a world with hundreds or thousands of rollups. It forces an average Ethereum user to deal with fragmented assets, forcing them to manage fragments of their assets across 100+ rollups along with complexities like GAS, bridging & RPCs for every transaction.

For example, today it takes over 30+ clicks in the wallet & over 40+ minutes

for a user to use an app on another chain with the swapping, bridging of tokens & GAS, switching RPCs, and other actions involved.

There are several standards today for smart contract wallets. Users send signed UserOps that are bundled together and sent on-chain. It also enables users to pay gas fees using ERC-20 tokens (e.g. USDC) instead of ETH by allowing a third party to sponsor their gas fees.

[

image

2000×799 75.5 KB

](https://ethresear.ch/uploads/default/original/3X/d/6/d696cf1819e42dfa3424b84e0c8360da38183915.jpeg)

The current smart wallet experience for users is not suited for transacting in a world with hundreds or thousands of rollups. It restricts the extent of abstraction to just one single rollup instead of thousands of rollups. It forces an average Ethereum user to deal with fragmented assets, forcing them to manage fragments of their assets across 100+ rollups along with complexities like bridging & RPCs for every transaction.

MagicSpend++

We propose MagicSpend++, a framework to allow users to magically spend on any chain instantly, without worrying about which chains their tokens are on.

MagicSpend++ leverages the existing Account Abstraction standard and builds on Coinbase's work on MagicSpend - an innovative approach to allow users to leverage their assets held on Coinbase Exchange & utilize them on-chain.

MagicSpend++ fundamentally enables users to have a Chain Abstracted Balance (CAB)

instead of isolated token balances across chains. Users can use their CAB to transact instantly on any chain with a signature. No fragmentation, no bridging, no GAS, no latency. It's magic.

Smart wallet users with MagicSpend++ get:

- A unified single balance across chains that users can spend anywhere
- Completely gas-less experience
- Instant single chain experience - NO CROSS CHAIN, NO BRIDGING, ZERO LATENCY

Fundamental Primitive — Time-locked Vaults

In order for tokens to enter the Chain Abstracted Balance, they need to be locked in their smart wallets. Users can withdraw funds back to their Externally Owned Accounts (EOAs)

after a configurable delay. While locked, the assets remain usable via the smart-account on any chain at any time. This one-time step required-to onboard as a part of deposit to SCW workflow, it is not required every-time before the assets are used.

All assets deposited represent the user's Chain Abstracted Balance (CAB). This chain abstracted balance can now be spent on any-chain via userOps, without bridging, just like how you would spend these assets as if they were on the chain you are acting on.

Your 100USDC@Optimism, 100USDC@Polygon, 100USDC@Arbitrum, 100USDC@Base, 100 USDC@ethereum all merge together into 400USDC, ready to be spent anywhere.

[

image

2000×857 113 KB

](https://ethresear.ch/uploads/default/original/3X/7/7/77765f9c7d128996817736249125fd859f3710f6.jpeg)

Key concept – Spend Now, Debit Later

Conceptually it's pretty similar to how AA works today with a small difference. User sends UserOp to perform an target chain, just like they would if they had funds on the same chain. Paymasters can now fund the userOp with not just gas but with additional funds to facilitate the on-chain execution of the UserOp. Paymaster then goes and claims it from the chain abstracted balance.

[

image

2000×1239 151 KB

](https://ethresear.ch/uploads/default/original/3X/8/5/85d1025c1eb22df5366dedd8449865387901327e.jpeg)

How it works - high level overview

Say Alice wants to purchase a NFT on BASE using her 1000USDC Chain-Abstracted-Balance:

- Alice sends a signed UserOp to purchase the NFT on BASE
- Paymaster checks(offchain) if there are sufficient funds with Alice in her chain abstracted balance and authorizes usage of paymaster funds via a special_signature
- UserOp is sent to bundler with paymasterAndData containing the special_signature that allows the SCW to pull funds from the paymaster's pool of liquidity on BASE
- UserOp is normally executed onchain leveraging the paymaster funds, SCW leverages withdrawGasExcess(special_signature) to pull funds from paymaster
- Paymaster can now debit the respective funds from Alice's chain abstracted balance in custody of smart-wallets (whichever chain or multiple-chains that they are on)

[

image

2000×920 130 KB

](https://ethresear.ch/uploads/default/original/3X/c/d/cdfdad93b81b35b2a546ea19f28e38c2990dd52d.jpeg)

Not Cross-chain but Chain Abstraction

This removes a lot of the pain a user faces today: figure out swap, bridge, buy new gas paying asset and only then will you get to use this new thing. This is dumb and holding back new innovation from taking off. A DEX can be 10x more capital efficient but without liquidity/volume it doesn't matter.

Chain abstracted wallets provide various 10x improvements for users:

- User asset-balances across chains are combined into one Chain Abstracted Balance
- Balance is instantly usable anywhere, zero bridging latency

This is completely different than any cross-chain implementations where wallets integrate bridges likes Across, Stargate, etc.

Cross-chain implementations are push-based where users start with a fragmented balance on one chain and "push" their tokens to another. On the other hand, MagicSpend++ is pull based where the user simply uses an app and their funds are pulled by the paymaster later. It's like Spend first, debit later.

This has various implications for the end users.

Cross-Chain Accounts (push)

Chain Abstracted Balance (pull)

Asset fragmentation still persists across chains. Users still have to think about fragmented assets on different chains

Single Chain Abstracted Balance

Bridging latency- Users sign a UserOp on the source chain, wait for source chain finality before tx is processed on target chain

Instant finality- User sign a UserOp on the target chain which is fulfilled immediately. No need to wait for finality on source chain.

Failure on dest chains - UserOp can fail mid-bridging. Users get stuck with arbitrary funds on target chain in this case

Guaranteed execution - user funds are deducted from the balance only if their transaction is successful

Users get tied into trust properties of the bridge for settlement

Users have tighter control over security where they can configure how paymasters can settle

Diving Deeper

All we really need to enable this is a new module that creates a vault with configurable withdraw delay such that users can deposit into this vault but only withdraw after a time period, the funds deposited are still usable by the user but in a chain-abstracted fashion. These vaults can be on multiple chains and all the balances will add up and be usable as one.

[

image

2000×1091 154 KB

](https://ethresear.ch/uploads/default/original/3X/2/9/299b49a7fa6fbf3619fbdc0c3b6cc846342ec7b9.jpeg)

A good example to start with is Alice wants to mint an NFT on Ethereum worth 100 USDC, lets assume the deposit into the SCW was already done on Base 1000 blocks ago

- Alice goes to the NFT mint site, and wants to insta buy
- Alice clicks mint button the paymaster service checks locked userBalance on all vaults across chains and sends back a special_signature
- paymasterAndData

****will be set to whatever paymaster service application/user opts into just like today, the data however includes the special_signature provided by the paymaster

- This userOp is sent to bundler → entrypoint
- Entrypoint calls validateUserOp() on the SCW and does the usual operations
- When Entrypoint calls validatePaymasterUserOp(……special_signature)
- Post the usual validations, the paymaster validates the signature and sets a map(address⇒amount) to allow userOp.sender withdraw the amount (which will be the SCW)
- The SCW will then leveraging the withdrawGasExcess

method pull funds from the paymaster and execute the userOp that buys the NFT

- In the postOp operation
- mapping(address⇒amount) set earlier will be deleted to ensure no double spends
- an execution proof will be sent by the sender contract on L1 to an L2 or multiple L2s via the native-rollup-connections
- execution proof is simply a proof to convince chainA of some execution that happened on chainB, think light-client proofs, ZKPs etc
- execution proof is simply a proof to convince chainA of some execution that happened on chainB, think light-client proofs, ZKPs etc
- mapping(address⇒amount) set earlier will be deleted to ensure no double spends
- an execution proof will be sent by the sender contract on L1 to an L2 or multiple L2s via the native-rollup-connections
- execution proof is simply a proof to convince chainA of some execution that happened on chainB, think light-client proofs, ZKPs etc
- execution proof is simply a proof to convince chainA of some execution that happened on chainB, think light-client proofs, ZKPs etc

- Funds will be then deducted from the userVault and given to paymaster

Interesting thing to note here, while the paymasters are forwarding funds to the user, they are not taking ANY reorg risk since there is no source chain tx for a userOp that is trying to spend, its credit now, debit later

Limitations and Solutions

While the above mechanism works great and accomplishes the desired goals, the approach has limitations with paymaster availability & settlement flexibility. Smart-Wallets can by-pass these limitations using Socket's MOFA.

- Single Paymaster Weak

: User gets locked into a single paymaster and if paymaster doesn't have liquidity userOps will not be executed. SCWs can instead delegate paymaster election to Socket's MOFA to tap into a global network of competing paymasters and sidestep liquidity and efficiency issues. With MOFA, multiple paymasters compete to fulfill incoming UserOps, which means more paymasters, more liquidity & more execution. Single paymaster weak, paymasters together strong.

- Restrictive Settlement Options

: Its quite important in the chain-abstracted world for users signing userOps or developers building SCWs to have the ability to expand to new chains quickly, to be able to make their own tradeoffs(security, cost, latency). Standardisation around settlement allows for users and developers to handle both expansion and tradeoff selection in a transparent fashion.

[

image

2000×1207 147 KB

](https://ethresear.ch/uploads/default/original/3X/3/5/35964d42692a8fd042edb63452bab73015cbf3c1.jpeg)

Acknowledgements

We think magic-spend++ enables complete chain-abstraction, where users dont have to care where their assets are.

Thank you to the following people for the many discussions, reviews and idea-sharing which lead to the creation of this proposal(in no particular order):

Wilson Cusack, Pierce Harger, Uma(@pumatheuma

), Alex Watts, Partha, Ahmed Al-Balaghi, Sachin Tomar, Ivo Georgiev, Theo Gonella, Derek Chiang, Kristof Gazso, Ankit and Stephane, Kakusan, Bapireddy, Aniket Jindal

We are keen to work with existing AA players building bundlers, paymasters and existing solvers/paymasters, wallets, dapps and together be at a place where no one needs to bridge anymore. If anyone is interested in magic-spend++ or chain-abstraction in general we have a public group here to talk about it: https://t.me/+QygeBurngS4wODNl

Account Abstraction → Chain Abstraction journey begins now

FAQs

- Are paymasters becoming solvers/fillers here?
- Solvers/Fillers in cross-chain setting for eg as used in Across, DLN and other intent based protocols are entities that front liquidity and take on reorg risk in return of fees, paymasters in the AA world are not needed to take on reorg risk and here paymasters arent exposed to reorg risk either, so no they arent the same
- Solvers/Fillers in cross-chain setting for eg as used in Across, DLN and other intent based protocols are entities that front liquidity and take on reorg risk in return of fees, paymasters in the AA world are not needed to take on reorg risk and here paymasters arent exposed to reorg risk either, so no they arent the same
- What risks/trust-assumptions are here for users?
- Users only need to trust the proof-system for security here and there is no additional off-chain security, magic-spend++ can leverage ZKPs, rollup-messengers to have an extremely trust-minimised setup
- Users only need to trust the proof-system for security here and there is no additional off-chain security, magic-spend++ can leverage ZKPs, rollup-messengers to have an extremely trust-minimised setup