EIP-7701

Native Account Abstraction

Why Account Abstraction?

Account Abstraction

- Smart Contracts can define validity of transaction payload - no EOAs involved
  - Passkeys
  - Session keys
  - Multisig signatures
  - Post-Quantum signatures
- Enable “gas abstraction”
  - Gas fees sponsored by 3rd party
  - Users charged with ERC-20
  - Privacy protocols withdrawal
- Execution is defined by the Smart Contracts’ implementation

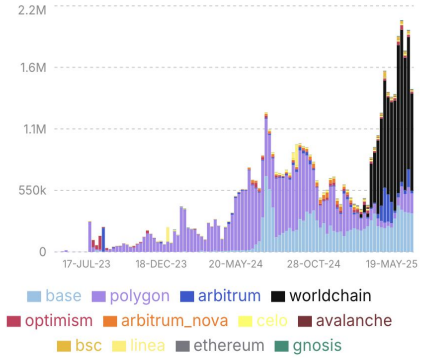
ERC-4337 Market Fit

ERC-4337 Metrics

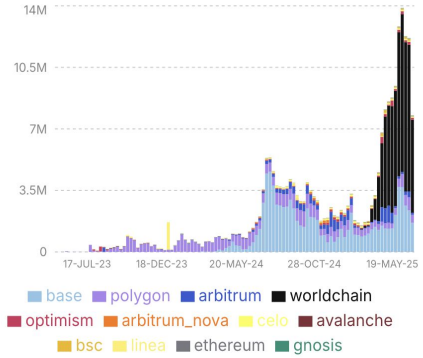
Total UserOps	Total Bundle Txns	Total Paymaster Volume	Accounts with 1+ Userops
238,994,323	**142,813,625**	**$6,702,668**	**32,531,327**
UserOps	Transactions	Gas Covered	Accounts

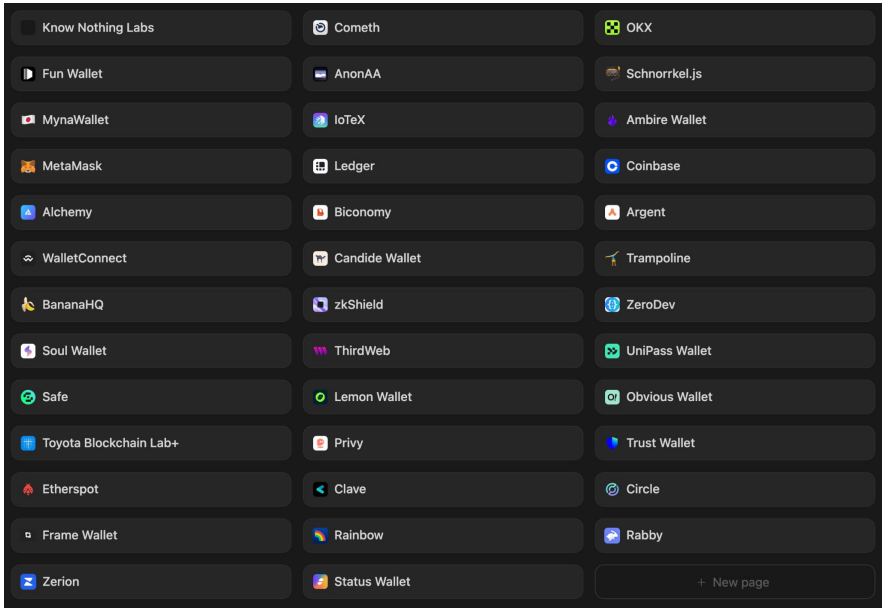
WEEKLY CHARTS

Weekly Active Smart Accounts



Weekly Sucessful UserOps





https://www.bundlebear.com/erc4337-overview/all

ERC-4337 Lessons Learned

- Enthusiastic adoption by wallets and dapps
  - Safe, OKX, Trust, Ambire, Coinbase, Soul, ZeroDev etc.
- Gas Sponsoring and Abstraction is a major case
  - 85% of UserOperations used a Paymaster
  - Sponsored operations are big source of adoption
  - Paymasters simplify new user onboarding
- Gas overhead hinders adoption
  - Using a private (centralized) relay is cheaper
  - Aggregated overhead is costly even on L2s

Paymaster	Total UserOps Served ↑↓	Total Gas Spend ↑↓
pimlico	40,638,145	$1,691,982.04
Unknown	24,038,969	$1,252,914.89
biconomy	17,277,178	$1,164,222.83
alchemy	119,509,841	$1,029,798.35
stackup	536,472	$815,192.31
coinbase	27,441,116	$748,894.62
particle	467,586	$77,622.27
circle	2,134,042	$15,709.37
send	909,379	$10,120.99
thirdweb	1,409,760	$8,223.88
candide	32,554	$2,344.19
ambire	98,539	$1,139.73
blocto	1,315	$1,018.45
cometh	108,176	$297.76
nani	482	$16.60
etherspot	378	$12.94

Why Native AA?

Native AA: Requirements

- Smart Contracts can define validity of transaction payload - no EOAs involved
- Enable “gas abstraction”: ETH paid by 3rd party, users charged ERC-20, privacy
- New Smart Contract can be deployed using Native AA without EOAs
- EIP-7870 hardware requirements are preserved for all nodes
- Avoid unnecessary complexity
- Avoid regressions

Native AA: Advantages

- Removes Gas Overhead
  - Gas overhead makes centralized relays more attractive
- Enables Censorship Resistance
  - FOCIL can support Native AA transactions
  - No need for centralized (censoring) “relay” or “bundler”
- Enables Post Quantum accounts
  - No need for PQ-EOA transaction type
  - No “migration” transaction - can use EIP-7702

Native AA: Gas Savings

ERC-4337:

- Static overhead: ~30’000 gas
- Per-operation overhead: ~40’000

This overhead appears to be inherent to the non-native approach.

EIP-7701 total overhead estimation: ~5’000

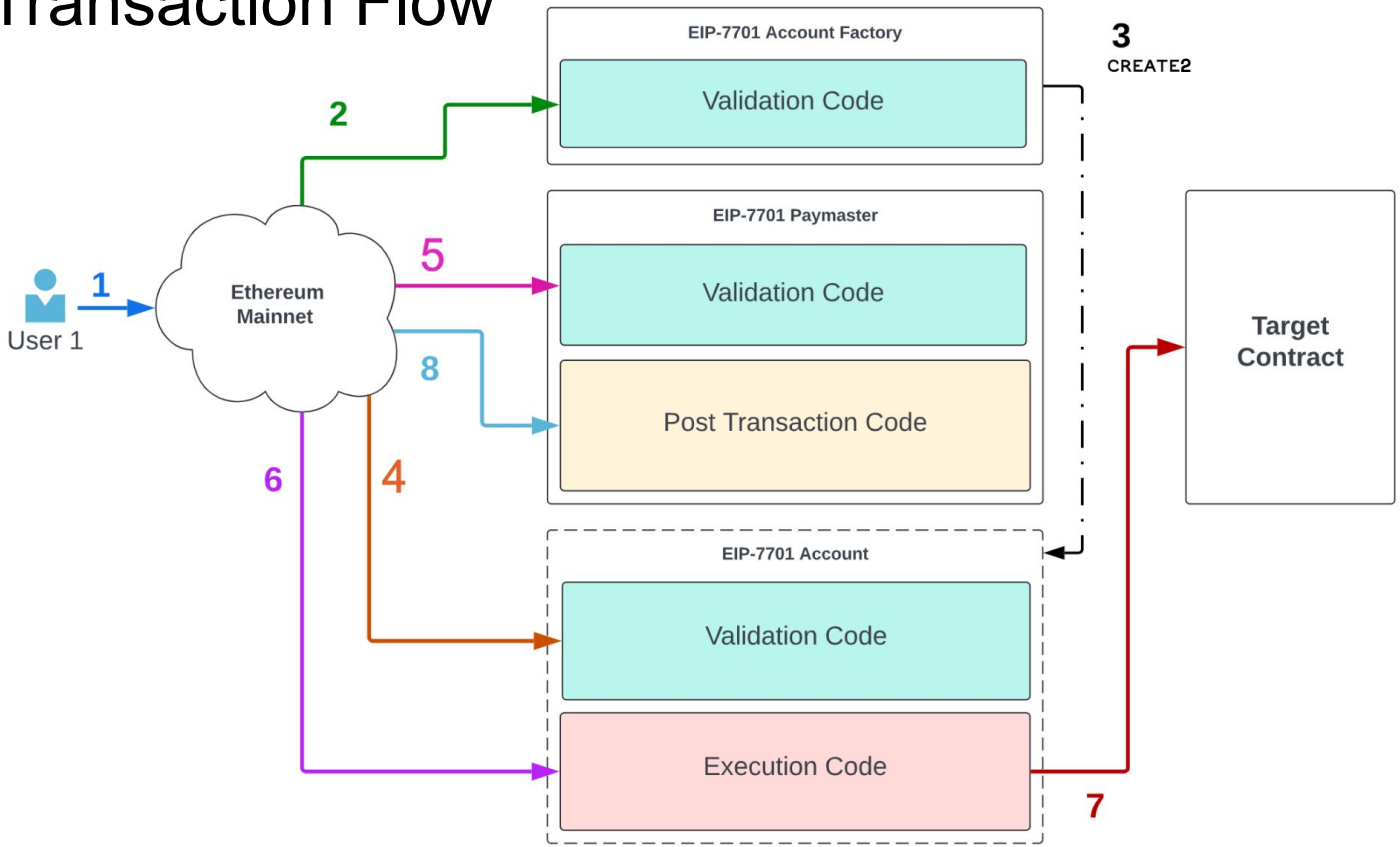
*measured for simple ERC-20 transfer - EOA vs ECDSA Smart Account

Why EIP-7701?

EIP-7701 Smart Contract Roles

- Sender Account
  - Validates the transaction
  - Pays the gas (if there is no Paymaster)
  - Executes the actual calls
- Deployer
  - Creates the Account (if necessary)
- Paymaster
  - Pays the gas (if specified)

EIP-7701 Transaction Flow



EIP-7701 Smart Contract Roles - why?

- Separating “validation” from “execution” required to prevent DoS on nodes
  - Limiting “validation” gas - a “potentially unpaid” work for nodes
  - Enable sandboxing the “validation” code to prevent N invalidations at O(1) cost
- Composability - Account and Paymaster are separate entities
  - Enables “mutual distrust”
  - Prevents potential Paymaster-vs-Account attack vectors

EIP-7701 Transaction Type

```
AA_TX_TYPE || rlp([
    chain_id,  nonce,
    sender, sender_validation_data,
    deployer, deployer_data,
    paymaster, paymaster_data,
    sender_execution_data,
    max_priority_fee_per_gas, max_fee_per_gas,
    sender_validation_gas, paymaster_validation_gas,
    sender_execution_gas, paymaster_post_op_gas,
    access_list,  authorization_list
])
```

EIP-7701 Opcodes

- **`TXPARAMSIZE, TXPARAMCOPY, TXPARAMLOAD`**
  - Available in the top-level frame
  - Provide access to all parameters of the **`AA_TX_TYPE`** transaction
- **`CURRENT_ROLE`**
  - Provides access to the identifier of the current role
- **`ACCEPT_ROLE`**
  - Indicates the calling contract accepts proposed role in the current **`AA_TX_TYPE`** transaction

EIP-7701 Transaction Validity

Protocol:

✅ Sufficient gas balance (account or paymaster)

✅ Correct account nonce

EVM Code Execution:

✅ Deployer accepted its role (first transaction for account)

✅ Account is deployed

✅ Account accepted its role

✅ Paymaster accepted its role (if paymaster specified)

EIP-7701 Transaction Mempool

- DoS possible if validation code has arbitrary state access
  - Mass Invalidation - i.e. one SSTORE invalidates multiple transactions
  - Becomes a big DoS vector
- Validation code limitations not part of core EIP - node configuration
- **ERC-7562** (not final) - set of limitations for validation code
  - Contracts only access “their own” storage
  - Ban environment introspection (time, blocks, gas, balances etc.)
  - Gas Limits for validation frames
  - Requires nodes to trace incoming EIP-7701 transactions’ validation frames

EIP-7701 with FOCIL and Statelessness

- FOCIL allows excluding IL transactions if invalid
- Checking validity of Native AA transaction consumes up to **`MAX_VALIDATION_GAS`**
- Checking validity of Native AA transaction requires bytecode and state
- Mitigations:
  - Reasonable limits on maximum total IL validation gas in AA-FOCIL
  - Reasonable limits on maximum witness sizes in AA-VOPS
  - Include contracts bytecodes in “active” storage for AA-VOPS

See the “Stateless” session on Friday!

EIP-7701: Alternatives

- Do nothing
  - AA remains ERC-4337 only
  - ERC-4337 UserOperations get no CR through FOCIL
    - Self-relaying possible but violates privacy
  - High gas overhead hinders adoption
  - For now, no solution for Post-Quantum accounts
- Make FOCIL aware of ERC-4337 UserOperations
  - Mixes application layer and protocol layer
  - All other issues mentioned above
- Another Native AA proposal supporting fewer use-cases

Extras

EIP-7701 Block building

- Block building and re-validation computation overhead
  - Each new cold SSTORE can invalidate only ONE transaction in mempool (ERC-7562)
  - Each mempool transaction can take up to **`MAX_VALIDATION_GAS`** to re-check
  - E.g. a transaction of 5’000’000 gas may cause **`1’000 x MAX_VALIDATION_GAS gas`** re-check
- Mitigations
  - Must select a **`MAX_VALIDATION_GAS`** very cautiously
  - Consider additionally limiting the number of SLOADs in the validation code
  - Consider using EIP-2930 ‘access lists’ enforcement in validation code
    - allows builders to delay re-checking affected transactions