The Future of EOF

Layer 1, Layer 2, and Beyond!

Danno Ferrin

Danno Ferrin

Director of Execution Verification and Assessment

Senior Consulting Principal Staff Engineer

What's in front of us, on the horizon, and over the horizon?

Will happen on Mainnet

Previously discussed and designed features targeting Post-EOF releases, but details need to be finalized.

May Happen on Mainnet

Requested features that align with the "EVM Endgame" roadmap and don't fundamentally alter the EVM.

Might Happen on Mainnet

Experimental features that need to be proven first,

or that provide L2 differentiation.

YOLO

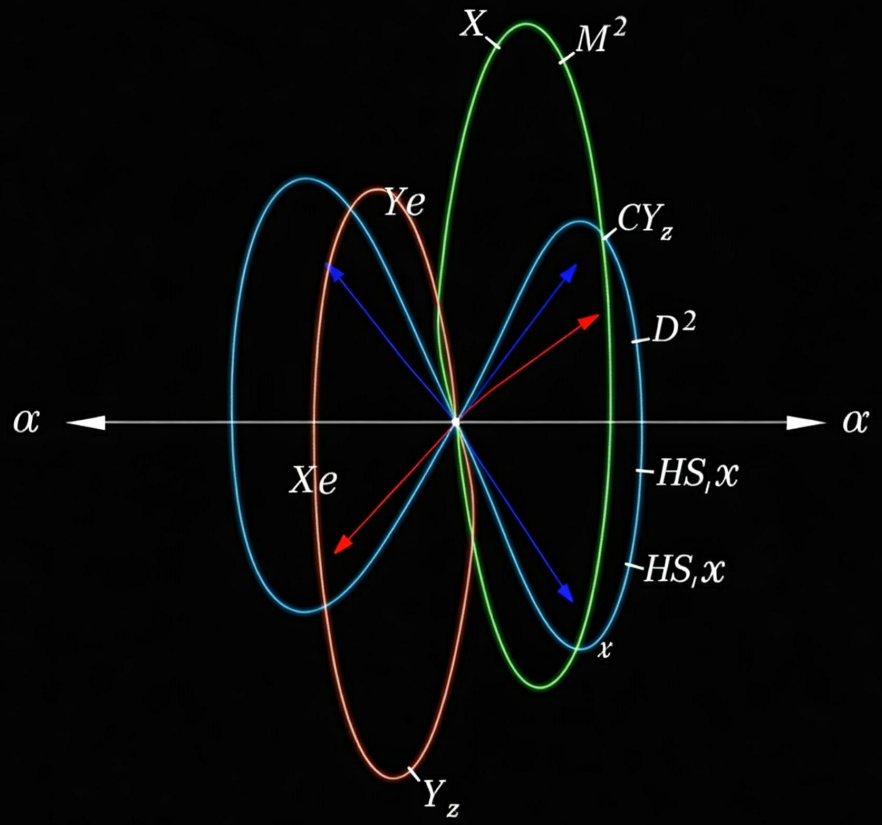
Will

What's in front

Reduce the need for precompile contracts

Precompile contracts present many problems for core development

- **They are bad for client code bases**
  - They introduce technical debt
  - They require specialized knowledge to optimize
- **They are bad for client security**
  - They increase the attack surface
    Gas attacks / DDOS / Client crashes / Incorrect implementation
- **They are bad for Project Management**
  - Champions don’t endure the ACD process well
  - Downstream projects can’t always wait



EIP-6690:
EVM Modular Arithmetic Extensions (EVMMAX)

Arbitrary size modular arithmetic

90-95% gas cost reduction

Essential math used in

- BLS-12-381
- MiMC / Poseidon
- Zk verifications

EIP-6690: EVM Modular Arithmetic Extensions (EVMMAX)

- Introduces 6 new Opcodes
- Introduces separate modular math memory space
- 1 opcode to init the space
  - Montgomery multiplication is the intended intermediate form
- 3 opcodes (ADDMODX, SUBMODX, MULMODX)
  - use immediate arguments to point into modular math memory space
- 2 opcodes to move final values in and out of MSTORE memory
  - Aligned on 64 byte boundaries

EIP-616:
SIMD Operations for the EVM

Single Instruction Multiple Data

Enables repeated math operations on large amounts of data

Useful for lattice cryptography

Probably too small for AI inference

EIP-616: SIMD Operations for the EVM

Proposes 25 vector operations for common math operations

Works with MSTORE memory

May see significant changes from current spec prior to implementation

Because most post-quantum crypto depends on lattice math,
SIMD support is inevitable

Proposal will almost certainly be heavily modified from current form.

Legacy / EOF Parity

Any legacy pattern not explicitly banned needs to be supported in EOF

- EOA account detection (ERC-721)
- ERC-4337 Entrypoint Contracts
- Popular Libraries like OpenZeppelin

Legacy to EOF Migration - Sunsetting Unvalidated Bytecode

Zero-Knowledge proofs work best with validated and predictable code free from code introspection.

EOF only chains are great, but mainnet has many legacy contract that the ecosystem depends on.

A multi-step process is needed to make mainnet SNARKable with as few special cases as possible.

Milestone: EOF / Legacy Parity
Have all features we plan to support available in EOF

Milestone: Legacy Deprecation
New contracts with "bad" features are not guaranteed to work.
New legacy deployments *may* be banned

Milestone: Mainnet Legacy Audit
All contracts prior to deprecation evaluated for "bad" features

May

On the Horizon

Enabling Better Mainnet Execution , Proven on Rollups

Enabling Account Abstraction

Native enablement will require execution environment integration,
EOF may make it nicer

Unbounded Code Size

Contract size has been frozen for 8 years
EOF validation fixes some issues

Concurrency

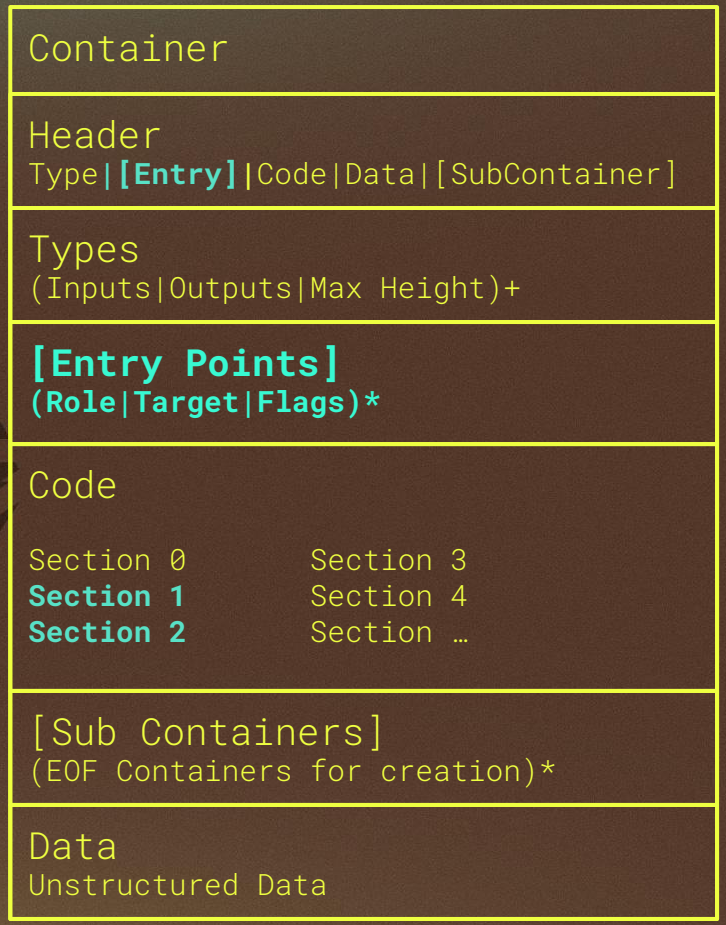
Parallelism is great, but concurrency is what delivers real utility.

EIP-7701: Native Account Abstraction with EOF

A variant of RIP-7560

Sets EOF code sections to handle validation and paymaster duties

Creates separate entry points from general execution



EIP-7701: Native Account Abstraction with EOF

`role_sender_deployment`

Once per account for setup

`role_sender_validation`

Authorizes based on signature, calldata, etc.

`role_sender_execution`

“Trampoline” into actual call, stack protection

(Paymaster, if used, is another contract)

`role_paymaster_validation`

Handles third party gas payments

`role_paymaster_post_tx`

Payment cleanup duties
(refunds, bookkeeping, etc).

Unbound Contract Sizes

Current size limit is 24KiB
    ~ a full block of gas in 2017

JUMPDEST Analysis kept it low
    Mitigated in Shanghai
    No in-protocol caching

EOF Validation is only at contract creation

640K ought to be enough for anybody.

Remark attributed to Bill Gates (Founder and CEO of Microsoft), 1981

The size limit will be increased before it is unbounded.

Contracts will still be gas limited.

(Bill probably didn’t say that)

EOF Limits with int16 and uint16

Code / Data / Subcontainer Sections
- 64KiB per byte size
- +/ 32Ki Jump target per bytes
- Max 1024 sections / containers

Max theoretical contract size
- 128MiB
  - 64MiB Code
    - 1024 sections x 64KiB
  - 64MiB Subcontainer
    - 1024 containers x 64KiB

Max practical contract size - 64KiB

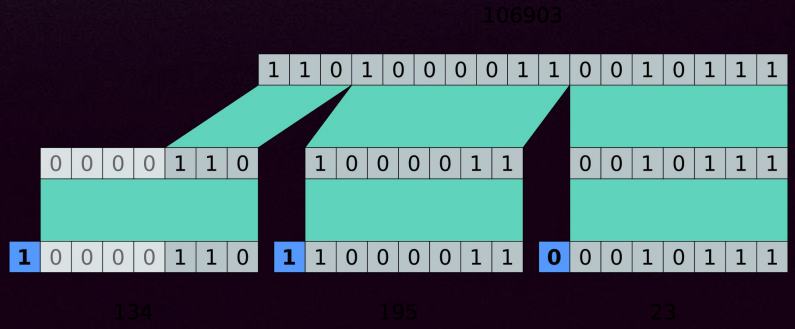
Optional Variable Length Quantity (VLQ) Headers and Immediates

Header and Immediate values could be VLQ encoded

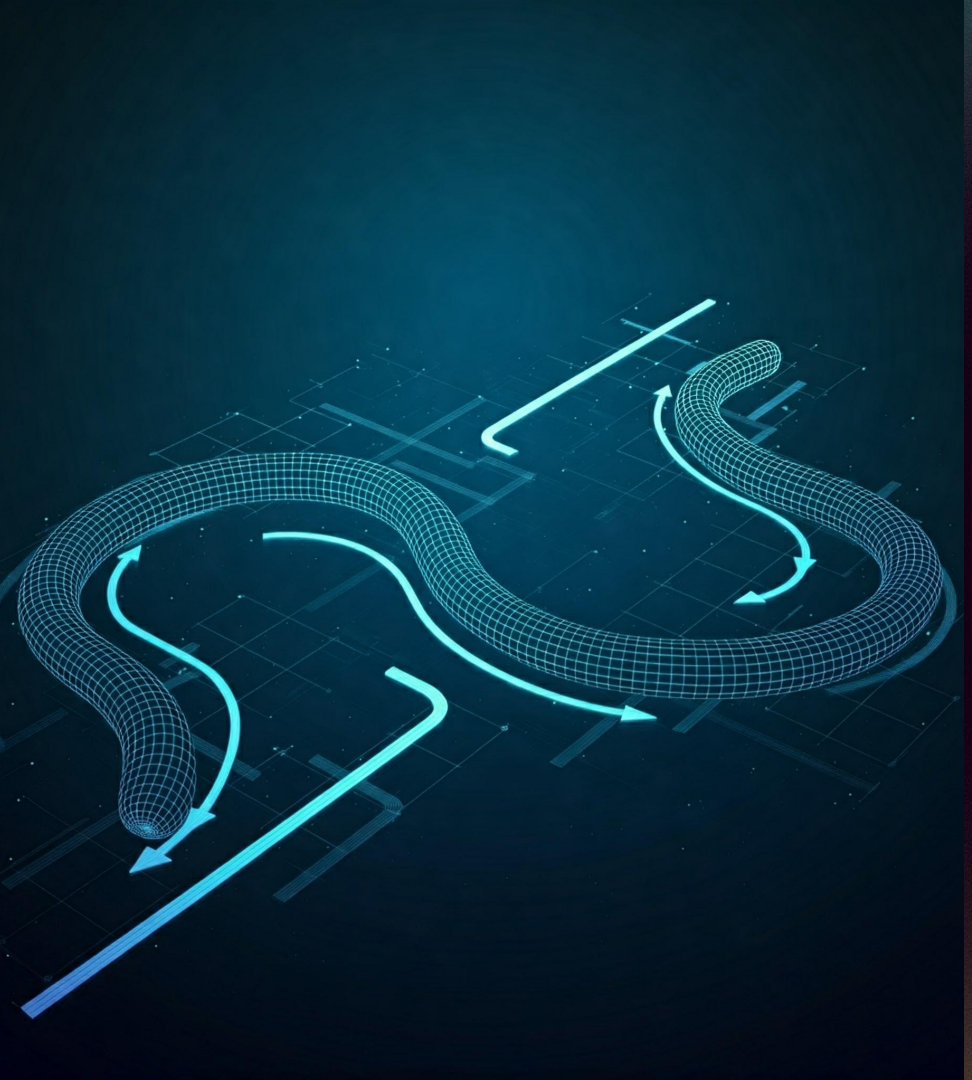
Smaller values see a code size reduction (less than 64 or 128)

Larger values see a code size increase

Gigantic values are possible



Potential downside: Full contract header must be streamed, making Verkle encoding complicated



Concurrency Support

Parallelism << Concurrency

Some concurrency will require EVM support

Atomic Operations

Express update as a delta

Deferred Execution

Run an additional transaction at the end of the block

EIP-7519: Atomic Storage Operations SCREDIT and SDEBIT

Two Opcodes added, to add or subtract from a storage slot

SCREDIT ~ DUP SLOAD ADD SSTORE
SDEBIT ~ DUP SLOAD SUB SSTORE

Value overflow and underflow is also checked, Halting the frame if uint256 under/overflow occurs

The expression of this operation *outside* the EVM is the real value.

SCREDIT(shemnon.eth, 100)

Instead of 500 -> 600 in witnesses
It can be +100 in witnesses

The balance of shemnon.eth can be masked in isolated proofs.

Queue End Of Block Transaction OPCODE

Proposes an opcode to queue up end of block transactions

Opens the door to TXes spawning other TXes, for async and “defer” semantics

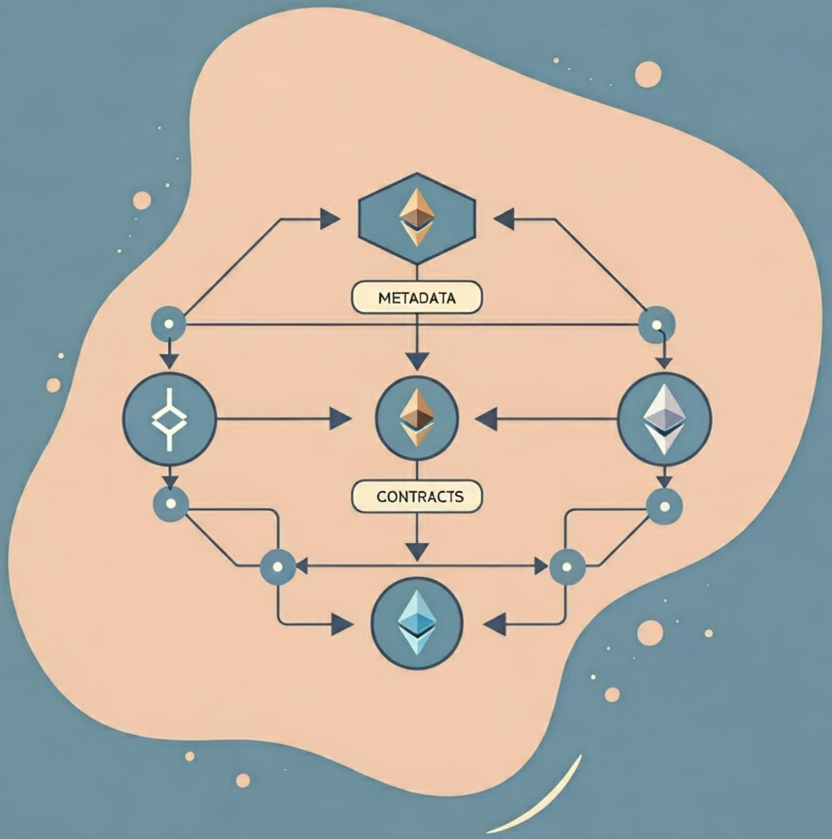
Isolates spawned transaction from parent transaction

ethresear.ch thread

Very early stage proposal

Might

Over the Horizon



Adventures in Meta-Data

Code and data are so last fork

What we need is data about the code
And data about the data

Data essential to use or execution

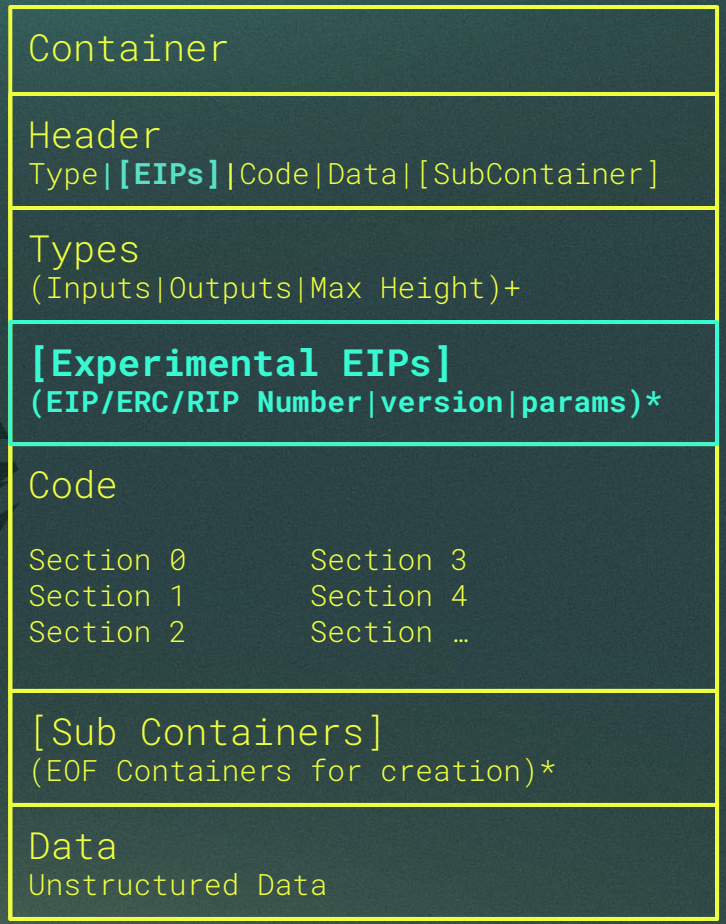
Encoding Experimental Feature Activations

Add a section enumerating optional EIPs activated.

Enumerate version activated

Parameters such as opcode numbers

L2s can determine at parse time if the features are supported.



Alternate Bytecode Formats / Optimized Binaries

EOF could contain non-evm bytecode

- WASM
- ARM
- RISC-V

Either as primary code or as optimized implementations

Progressive precompiles via CREATE2 shadowing

True precompiles

- An EVM version of the logic that provides the canonical result
- Clients could optimize it so long as the optimized version returns identical results
- Address can be derived from standard CREATE2 factories, pre-deploy in genesis or deploy at runtime

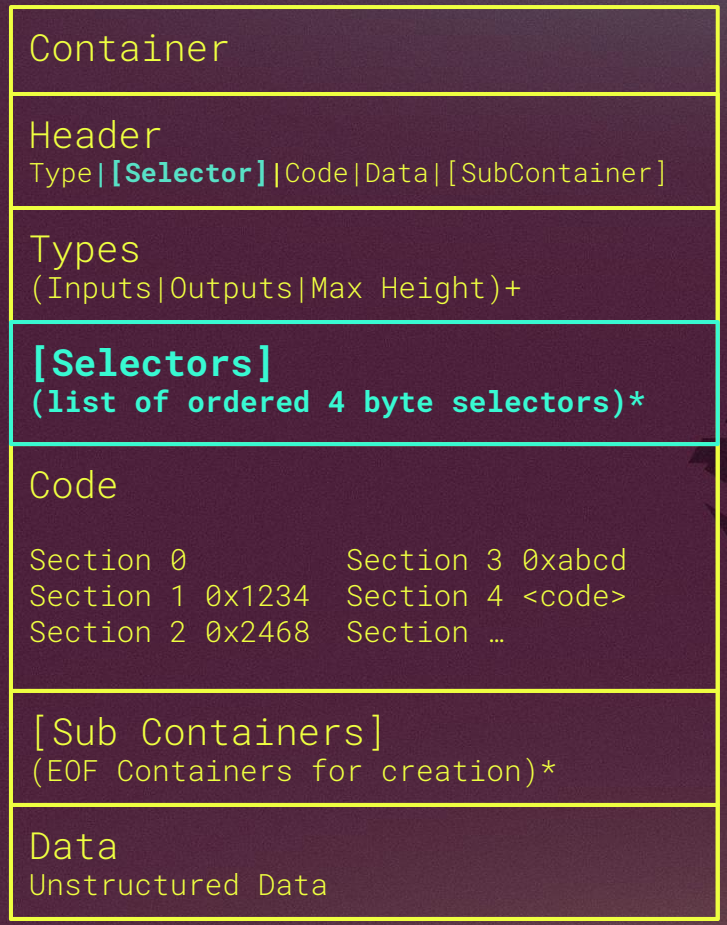
Proposed in an ethereum-magicians.org thread

Thread relies on legacy EVM and legacy CREATE2 deployer, EOF facilities would need to be created

Alternate Gas Calculations

A variation on EIP-7701.

A role would provide the gas to charge as a function of the inputdata.



Optimized Solidity Dispatch

A variation on EIP-7701

Map entrypoints to the first 4 bytes of call-data

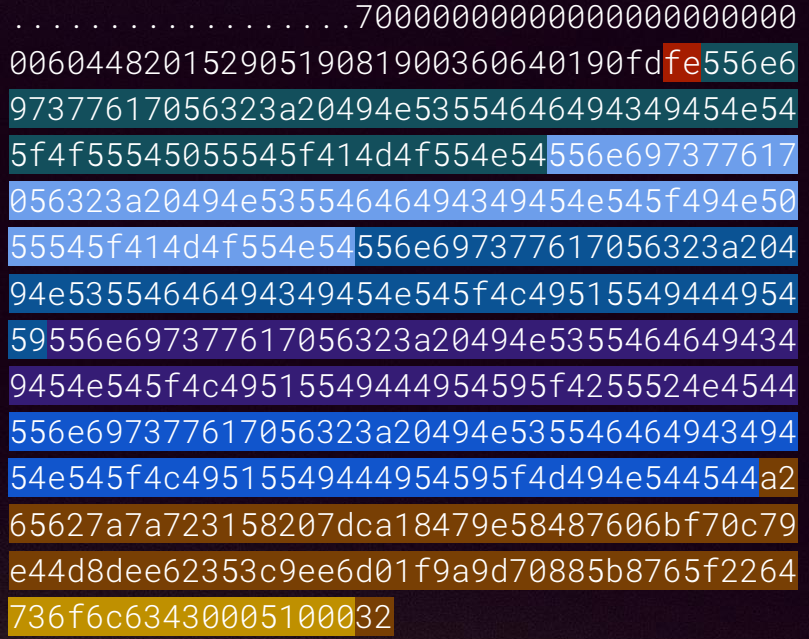
Section 0 is always fallback for short call data or missed matches

Embed Compiler Metadata into the EOF Header

Solidity embeds multiple kinds of data organically from the source

- Revert and error names
- Hash of all of the sources
- Version of the solidity compiler

Instead of packing it into data some metadata could live as EOF headers



Summary: What Will Be, Could Be, and Might Be the Future

On mainnet one day

EIP-6690 - EVMMAX, Better Modular Math

SIMD instructions, Once lattice encryption is the rage

EOF Parity / Legacy Deprecation

Things to Watch for

EIP-7701 - Account Abstraction Integration

Larger contract sizes, Maybe only gas limited!

EVM support of Concurrency Initiatives

Metadata Possibilities

Experimental Activations

Alternative Bytecode

Gas Schedule Definition

Enshrined Dispatch

Thank you!

Danno Ferrin

Senior Consulting Principal Staff Engineer
Verification and Assessment, Numisight
danno.ferrin@gmail.com
@shemnon