In the suave-specs there is currently some discussion on how to enable multi-chain support for SUAVE. I have summarized the conversation so far below in hopes of bringing more visibility to the idea space.

Problem Statement

- The current SUAVE implementation is limited to interacting with a single, Dencun Compatible L1 blockchain at a time, restricting its operation in a multi-chain context.
- There's a need to generalize the execution namespace within SUAVE to enable execution calls to nodes on any chain supporting this namespace.

Design Goals

- Multi-Chain Compatibility:

Enable SUAVE to seamlessly interact with multiple blockchain networks whose nodes support the execution namespace.

- Flexible Payload Crafting:

Develop a mechanism within SUAVE for efficiently crafting and managing payloads for different chains.

Challenge

- Ensuring a SUAPP is aware of the execution services a kettle exposes is challenging, especially when dealing with encrypted CCRs and the need for smart contracts to craft payloads flexibly using JSON RPC requests through the doHttpRequest

precompile.

Proposed Solutions and Discussions

- Multichain Compatibility:
- Suggestion to allow suave.eth.remote_endpoint

to support comma-separated strings in the format chainId:rpc_url

, enabling SUAVE to reroute remote RPC requests to the appropriate URL based on the chain ID.

- Modification of the buildEthBlock

precompile to include a chain ID parameter, enabling users to specify the destination chain for block construction.

- Suggestion to allow suave.eth.remote_endpoint

to support comma-separated strings in the format chainId:rpc_url

, enabling SUAVE to reroute remote RPC requests to the appropriate URL based on the chain ID.

- Modification of the buildEthBlock

precompile to include a chain ID parameter, enabling users to specify the destination chain for block construction.

- Flexible Payload Crafting:
- Generalizing the buildEthBlock

precompile to remove the need for specifying buildBlockArgs

, with the payload attributes being handled by the RPC remote backend.

- The necessity to modify sequencers for chains with a single trusted sequencer to share payload attributes.

- Generalizing the buildEthBlock

precompile to remove the need for specifying buildBlockArgs

, with the payload attributes being handled by the RPC remote backend.

- The necessity to modify sequencers for chains with a single trusted sequencer to share payload attributes.

- Challenges and Considerations:

- Whether all kettles need to connect to multiple chains and the social consensus on-chain ID to chain mapping.
- A different approach suggests deprecating certain endpoints in favor of building functionality using Solidity and the arbitrary HTTP precompile in suave-std as another protocol.
- The idea of a name resolve service on the MEVM for the arbitrary HTTP endpoint, allowing for more flexible and secure endpoint specifications.
- Whether all kettles need to connect to multiple chains and the social consensus on-chain ID to chain mapping.
- A different approach suggests deprecating certain endpoints in favor of building functionality using Solidity and the arbitrary HTTP precompile in suave-std as another protocol.
- The idea of a name resolve service on the MEVM for the arbitrary HTTP endpoint, allowing for more flexible and secure endpoint specifications.
- Security and Extensibility Concerns:
- Concerns about allowing arbitrary URLs in contracts due to potential leaks of confidential information.
- Proposals for a DNS-like approach to offer a more extensible solution than chain ID mapping, potentially allowing multiple RPC endpoints per chain.
- Concerns about allowing arbitrary URLs in contracts due to potential leaks of confidential information.
- Proposals for a DNS-like approach to offer a more extensible solution than chain ID mapping, potentially allowing multiple RPC endpoints per chain.