Interoperability explainer

Interoperability is the ability for a blockchain to securely read the state of another blockchain. Native OP Stack interoperability provides the ability to read messages and transfer assets across the Superchain (without having to go through L1) via secure message passing. This results in the following benefits:

- fungible and portable assets and liquidity
- lower fees and lower latency
- less fragmentation across the Superchain
- improved user experience for developers on the Superchain
- secure transfer of ETH and ERC-20s across L2s

Secure message passing

Superchain interop includes both the protocol layer message passing and the Superchain ERC20 token specification.

- Message passing protocol:
- the initial + finalizing/executingmessage
- that fire events to be consumed by the chains in thedependency set(opens in a new tab)
- SuperchainERC20 token specification
- : theSuperchainERC20
- turns message passing into asset transfer between chains in the interop set. Learn more about how the SuperchainERC20 token standard enables asset interoperability in the Superchainhere

This means ETH and ERC-20s can seamlessly and securely move across L2s, and intent-based protocols (i.e., bridges) can build better experiences on top of the message passing protocol.

Low latency

Interoperability allows for horizontally scalable blockchains, a key feature of the Superchain. With Superchain interop, latency can be low (~2 seconds) by optimistically accepting cross-chain messages. The fork choice rule enforces eventual consistency, meaning that if an invalid cross-chain message is accepted, it will be reorganized out eventually. The fault proof guarantees that all of the cross-chain messages are accounted for from the perspective of handling withdrawals through the bridge to L1.

Permissionless chain set

It is permissionless to define a dependency on a chain, so chain operators will be able to choose which chains their chains depend on, and it is not a requirement that chains are in each other's dependency set. Chain operators can add or remove chains from the dependency set through theSystemConfig . For example, the dependency set for OP Mainnet is managed by Optimism Governance.

However, since the nature of defining a dependency is one way, it is impossible for a chain to know of all of the other chains that depend on it.

Considerations

Chain operators will need to run additional infrastructure to be part of the interoperable set.

- The Superchain-backend service,op-supervisor
- , will be a requirement for running an OP Stack chain that has interop enabled.op-supervisor
- is responsible for validating all cross-chain messages and will need to have an RPC configured for each chain in the dependency set.
- In the future, to reduce infrastructure costs,op-supervisor
- will rely on the P2P network and cryptographic schemes for validating cross-chain messages.

For additional considerations, join theInterop discussion(opens in a new tab) in our specs repo.

Next steps

- Want to learn more? Read our guide on the anatomy of across-chain message
- or check out thisinterop design video walk-thru(opens in a new tab)
- .
- Ready to get started? UseSupersim
- , a local dev environment that simulates interop for testing applications against a local version of the Superchain.
- For more info about how OP Stack interoperability works under the hoodcheck out the specs(opens in a new tab)

- .

FAQs

What is the latency/security tradeoff?

If a sequencer does not want to trust another sequencer at all, they can choose to only accept executing messages where the initiating message has been finalized with L1 levels of security. This demonstrates the difference in latency vs speed. The L2 data must be submitted to L1 and then the L1 block containing the transaction with the L2 data must be finalized. It takes about 15 minutes for an L1 block to finalize, and higher throughput OP Stack chains like Base and OP Mainnet submit a batch about every 5 minutes. This means that it could take about 20 minutes for an L2 transaction to be considered final and be able to be trustlessly consumed by another chain. At lower latencies, the sequencer accepting the executing message takes on some amount of equivocation risk. It could be possible to build a bonding system to add economic security to prevent equivocation, but that is not in the scope of the first release.

See thisdune dashboard(opens in a new tab) to see how often OP Stack chains submit batches.

What is stopping a sequencer from censoring a cross-chain message?

There is nothing stopping a sequencer from censoring a transaction when it is sent directly to the sequencer. This does not mean the network has no censorship resistance, users can always send a deposit transaction for censorship resistance as strong as L1 guarantees. The tradeoff here is the latency, instead of being confirmed in ~2 seconds, the transaction can be confirmed at the rate of L1 block production. It may be possible to adopt something likeEIP-7547(opens in a new tab) in the future to enable low latency censorship resistance.

Are callback style transactions possible?

If two blocks are being built at the same time with shared knowledge of their contents, it is possible to build blocks where a transaction calls to another chain, does compute and then a transaction calls back with the results. This requires no protocol level changes, it just requires more sophisticated block builder infrastructure.

What is stopping a shared sequencer from including just the executing message and not the initiating message?

The protocol enforces the fact that all executing messages are valid. It does this by reorganizing out executing messages that have invalid initiating messages. Running software that does not enforce this would be non-standard behavior and would leave yourself at risk of accepting an invalid executing message and therefore running on a forked chain.

What is the trust/verification model? Do sequencers that opt into this interop system have to trust each other fully?

Sequencers only have to trust each other, if they are accepting executing messages where the initiating message is unsafe. This is because the sequencer's ability to equivocate on unsafe data, i.e., batch submit something different from what they gossip over the p2p network. Once data is submitted to L1, it is considered final relative to the L2 and therefore there is no longer an equivocation risk.

Is interop different between chains with non-fungible blockspace?

Chains that have non-fungible blockspace are chains that have different features - it could be that they use Plasma for data availability, a custom gas paying token or have a large execution gas limit. As long as the chain can be fault proven, it can work with Superchain interoperability. At the application layer, it is important for chains to have legibility into the type of chain that the message originated from. This ensures that applications do not accept messages from chains they consider not secure enough. See thisdiscussion(opens in a new tab) for additional thoughts.

When it comes to chains that have different gas limits that are interoperable, it creates a set of transactions that can execute on one chain but not the other. This happens when the transaction consumes more than the gas limit of the smaller chain but less than of the bigger chain. At 2024 usages levels, these sorts of transactions are very rare. In the future, it may be the case that these transactions become more common, it will be up to the chain operators to ensure quality of service for their users.

Interoperability Anatomy of cross-chain message