Using the Espresso Network

Guides for applications, such as rollups and bridges, to use the Global Confirmation Layer.

Background

The Espresso Network is a shared source of truth that provides strong confirmations on transaction ordering and data across chains. The confirmations provided by HotShot are additive, meaning that rollups can keep giving their users pre-confirmations using their own existing sequencer. It's up to the end user whether to trust the centralized sequencer or wait a few seconds more for the stronger confirmation provided by Espresso. For more details see the system overview docs. .

Using Espresso confirmations

Using the Espresso Network primarily relies on three ingredients:

- Data is sent to Espresso
- Data is fetched from Espresso after confirmation
- HotShot consensus is verified in the chain's state transition function (STF)

We recommend running an Espresso node and using its query service. This is the easiest way to ensure that all data received is validated. For convenience you can also connect to the API services of Espresso's nodes and use the espresso-dev-node Docker image for local testing.

Sending data to Espresso

Espresso does not interpret the data submitted to the network. As a result, chains are free to encode their data however they wish when sending it to Espresso. Chains are expected to choose an integer namespace ID akin to the EVM chain ID. Note that anyone can submit transactions with any namespace ID. If transactions from third parties are undesired those must be filtered out.

An Espresso transaction consists of a namespace ID and payload bytes.

The Espresso transactions can be sent to the submit endpoint of sequencer nodes, or to a builder. Builders may support private mempools if gossiping of transactions through the public mempool is not desired.

Fetching data from Espresso

The blocks finalized by Espresso are divided by namespace. Transactions with a certain namespace ID will be found in the corresponding namespace in the block. For a basic integration, the rollup will only consider the Espresso transactions in its own namespace.

Receiving notification about finalized Espresso blocks

Rollups can receive notifications from the Espresso query service API or the sequencer contract on the L1. The interval at which the finalized state is proved in the sequencer contract (hours) is orders of magnitudes slower than the Espresso block time (seconds). To acquire confirmations, notifications about newly finalized blocks must be obtained via the Espresso query service API directly.

Fetching transactions confirmed by Espresso

Confirmed transaction data can be obtained from the Espresso DA layer. The query service of Espresso nodes provides a variety of API endpoints to fetch finalized blocks, namespaces, and transactions.

Verifying Espresso consensus

The rollup state transition function must ensure that the rollup transactions are confirmed by Espresso before executing them.

The two main additions to the rollup STF are that the STF must verify that:

1. the transactions are exactly the transactions in the rollup's namespace confirmed as part of the finalized Espresso block, and
2. the Espresso block is part of the canonical Espresso chain.

The first check involves verifying Espresso DA's Savoiardi VID proof. To learn more about our VID scheme refer to our DA documentation or Appendix A of the Espresso Paper.

For the second check, the STF needs to verify the Espresso block Merkle proof and compare that against the proven block Merkle tree commitment in the Espresso light client contract on the L1.

The STF must also ensure that no transactions finalized by Espresso are skipped or applied twice.

Arbitrum Nitro integration example

Espresso's integration with the Arbitrum Nitro stack provides an illustration of how Espresso consensus can be verified in an application's STF.

If the architecture of the rollup mandates that all data is available on a specific DA layer, the integration needs to ensure the necessary data is submitted to that DA layer. In the Espresso-Arbitrum Nitro integration, a *Justification* (sometimes called *Attestation*) contains the namespace and block Merkle proofs necessary for validation of Espresso consensus. The justification is submitted to the Nitro parent chain so that it can be used as part of the STF to derive the state of the child chain.

The Espresso-specific validation for the Arbitrum Nitro STF is performed in the replay binary here. It calls out to the Rust code to verify a namespace and to verify the Espresso block Merkle proof.

In the Arbitrum Nitro "Arbitrator" smart contract, two additional functions added to the Host IO contract are used to read and validate the HotShot/Espresso commitment from the sequencer contract. Due to Nitro's stateless STF, the integration also adds the corresponding Host IO functionality to supply this external piece of data to the STF when validator and staker nodes execute them locally.

Integrating Arbitrum Orbit Chain

Arbitrum integration with Espresso

Espresso has developed an integration with the Arbitrum Nitro tech stack that allows Arbitrum Orbit chains to easily integrate with Espresso. The first version of this integration enables Espresso confirmations for Orbit chains. The integration will later be updated with functionality to enable enhanced cross-chain interoperability, new forms of sequencing (i.e., decentralized/shared sequencing), and support for Espresso DA.

The first production release of the Espresso-Arbitrum Nitro integration is planned for January 2025, at which point Arbitrum Orbit chains will simply be able to download a Docker image that allows them to deploy onto Espresso (or ask their RaaS provider to do so on their behalf).

If you just want to see the steps for getting up and running, you can skip to the guide.

Integration overview

This integration makes minimal changes to the Arbitrum Nitro stack, and ensures that each batch processed by the rollup is consistent with HotShot-finalized blocks within its namespace.

To ensure that the batch has been finalized by HotShot, the following checks are performed:

1. **Namespace validation:** Ensure that the set of transactions in a rollup's batch corresponds to the correct namespace. Namespacing allows multiple chains to use Espresso's fast confirmation layer simultaneously by associating each chain's transactions with a unique namespace within HotShot blocks.
2. **Espresso block Merkle proof check:** Confirm that the rollup's batch maps to a valid HotShot block. Specifically, verify that the HotShot block associated with a rollup batch is a valid leaf in the Merkle tree maintained by the light client contract, which stores the state of HotShot on L1.

High-level flow of integration

1. The sequencer calls WriteSequencerMsg on the transaction streamer.
2. The batcher fetches the message from the transaction streamer and submits the transaction to HotShot via the transaction streamer.
3. The batcher then calls the query API to check if the transaction has been finalized by HotShot.
4. Once the transaction is finalized, the batcher performs batch consistency checks.
5. The batcher signs the transaction calldata that would be sent to the Sequencer Inbox contract.
6. The Sequencer Inbox contract is modified to verify the batcher's signature.

This approach involves running a Nitro node with only the batcher enabled, operating in a TEE environment (such as Intel SGX). The batcher will sign the transaction calldata. In case the TEE is broken, the batch poster can't impact the safety of the Orbit chain. It could, however, temporarily halt the chain's progress, thereby breaking liveness. Bridges relying on Espresso confirmations for faster settlement need to trust the TEE as well in this integration. In a future update, the dependency on TEEs will be removed entirely.