L2 to L1 messaging

Arbitrum 'sOutbox system allows for arbitrary L2 to L1 contract calls; i.e., messages initiated from L2 which eventually resolve in execution on L1. L2-to-L1 messages (aka "outgoing" messages) bear many things in common with Arbitrum'sL1-to-L2 messages (Retryables), "in reverse" though with a few differences.

Protocol Flow

Part of the L2 state of anArbitrum chain — and consequently, part of what's asserted in everyRBlock — is a Merkle root of all L2-to-L1 messages in the chain's history. Upon an asserted RBlock being confirmed (typically ~1 week after its asserted), this Merkle root is posted on L1 in theOutbox contract. The Outbox contract then lets users execute their messages — validating Merkle proofs of inclusion, and tracking which L2 to L1 messages have already been spent.

Client Flow

From aClient perspective, an L2 to L1 message begins with a call to the L2ArbSys precompile contract'ssendTxToL1 method. Once the message is included in anAssertion (typically within ~1 hour) and the assertion is confirmed (typically about ~ 1 week), any client can execute the message. To do this, the client first retrieves the proof data via a call to the Arbitrum chain's "virtual"/precompile-esque**NodeInterface contract'sconstructOutboxProof method. The data returned can then be used in theOutbox 'sexecuteTransaction method to perform the L1 execution.

Protocol Design Details

An important feature in the design of the Outbox system is that calls toconfirmNode have constant overhead. Requiring thatconfirmNode only update the constant-sized outgoing message root hash, and that users themselves carry out the final step of execution, achieves this goal; i.e., no matter the number of outgoing messages in the root, or the gas cost of executing them on L1, the cost of confirming nodes remains constant; this ensures that the RBlockConfirmation processed can't be griefed.

Unlike Retryables, which have an option to provide Ether for automatic L2 execution, outgoing messages can't provide in-protocol automatic L1 execution, for the simple reason that Ethereum itself doesn't offer scheduled execution affordances. However, application-layer contracts that interact with the Outbox could in principle be built to provide somewhat-analogous "execution market" functionality for outsourcing the final L1 execution step.

Another difference between outgoing messages and Retryables is that Retryables have a limited lifetime before which they must be redeemed (or have their lifetime explicitly extended), whereas L2 to L1 messages are stored in L1 state, and thus persist permanently / have no deadline before which they must be executed. The week long delay period before outgoing messages can be executed is inherent and fundamental to the nature of Arbitrum Rollup, or any Optimistic Rollup style L2; the moment aTransaction is published on-chain, any observer can anticipate its result; however, for Ethereum itself to accept its result, the protocol must give time for Arbitrum validators to detect and prove fault if need-be. For a protocol overview, seeHow Arbitrum works

** We refer toNodeInterface as a "virtual" contract; its methods are accessible via calls0x00000000000000000000000000000000000000C8 , but it doesn't really live on chain. It isn't really a precompile, but behaves a lot like a precompile that can't receive calls from other contracts. This is a cute trick that let's us provide Arbitrum-specific data without having to implement a custom RPC. Edit this page Last updatedonJan 27, 2025 Previous L1 to L2 messaging Next A gentle introduction