Overview: The Lifecycle of an Arbitrum Transaction

As an introduction to the various components that compose the Arbitrum protocol, we'll go step-by-step over the phases an Arbitrum transaction goes through, starting with a client creating a signed transaction, to it ultimately being confirmed back on layer 1.

We'll also intersperse it with "finality checks," explaining what guarantees the client has over their transaction's finality (i.e., assurances that their transaction's result is guaranteed and won't later be altered) over the course of a transaction's various stages.

This overview will be focused on the Arbitrum Rollup protocol; seeInside AnyTrust for differences in the Arbitrum AnyTrust protocol. Also, for convenience/simplicity, we'll be describing the security properties of the core system itself; see"State of Progressive Decentralization" for current decentralization status.

For clarity on any terminology that may be unfamiliar, see ourglossary .

1. Sequencer receives transaction

Typically, a transaction's lifecycle starts with the Sequencer, the entity designated with transaction ordering, receiving a transaction from a client. The Sequencer can receive a transaction one of two ways:

1a. Directly / Offchain

For typical transacting within the L2 environment (i.e., using an L2 native dapp), a client will connect their wallet to an L2 node and directly deliver a signed transaction.

1b. ... or from L1 (via the Delayed Inbox).

Alternatively, a client can send a message to the Sequencer by signing and publishing an L1 transaction in the Arbitrum chain's Delayed Inbox. This functionality is most commonly used for depositing ETH or tokens via a bridge.

See :

- Retryables
- The Sequencer
- Token Bridge

2. Sequencer orders transaction (off-chain)

Upon receiving a transaction, the Sequencer will:

- Order it in its off-chain Inbox
- Locally execute it using the Arbitrum Nitro VM (including collecting/allocating L1 and L2 fees, etc.)
- "Instantly" give a transaction receipt to the client ("instant" in that it doesn't require any additional on-chain confirmations, and typically takes less than a second - with the average user experiencing ~260ms).

See :

- ArbOS
- Geth
- L1 pricing
- /L2 Gas

~ ~ ~ FINALITY CHECK: Trusted / Soft Confirmation ~ ~ ~

At this phase, the client's acceptance of finality relies on trusting the Sequencer. I.e., a malicious/faulty Sequencer could deviate between what it promised in the transaction receipt and what is ultimately published in a batch (see phase 3).

note Even a malicious/faulty Sequencer can only, at worst, reorder or temporarily delay transactions; it cannot, e.g., forge a client's transaction or propose an invalid state update. Given the degree of trust in the Sequencer at phase 2, we sometimes refer to the "instant" receipt that the Sequencer provides as a "soft confirmation."

3. Sequencer posts transaction in a batch (on-chain)

The Sequencer will eventually post a batch of L2 transactions which includes our client's transaction onto the underlying L1 (as calldata); under normal conditions, the Sequencer will post batchesevery few minutes .

3a. What if the Sequencer never includes our transaction?

Even if the Sequencer never includes our transaction in a batch, the client can include it in the L2 by posting in the delayed inbox and then "force including" it after some delay period (currently ~ 24 hours on Arbitrum One).

note The Sequencer is forced to include messages from the delayed Inbox in the queued order that they appear on chain, i.e. it processes messages using the "first in, first out" method. Thus, it can't selectively delay particular messages while including others; i.e., delaying the message at the front of the queue means delaying all messages behind it as well. See:

- "The Sequencer / Censorship Resistance."

~ ~ ~ **FINALITY CHECK: Ethereum-Equivalent Finality!** ~ ~ ~

At this stage, assuming that a client believes there to be at least one well behaved active Arbitrum validator. Currently, the process of validation on the Arbitrum protocol is permissioned, but it's important to be aware that our latest dispute protocol,BOLD (Bounded Liquidity Delay) , has the potential to allow validation on Arbitrum chains without requiring permission, thereby potentially eliminating the necessity for restricted validation. The client can treat their transaction's finality as equivalent to an ordinary Ethereum transaction. In other words, their L2 transaction has the same finality as the L1 transaction that recorded it in a batch. This means the client should use whatever finality heuristic they use for regular Ethereum transactions (i.e., waiting on L1 block confirmations, etc.), applied to the L1 batch-posting transaction. This also means that a client uncomfortable with the trust model of the Sequencer's soft confirmations (phase 2) can simply wait for the Sequencer to post their transaction in a batch (phase 3).

How are we able to make such bold a claim? A few (related) things:

- Once the Sequencer posts a batch, its transactions' ordering is entirely determined by the L1; the Sequencer effectively has no more say in our transaction's lifecycle at all.
- The Inbox contract on L1 ensures that when the Sequencer posts a batch, it posts data sufficient for any Arbitrum Node to reconstruct and validate the state of the L2 chain; i.e., the availability of this "input" data is guaranteed by Ethereum itself.
- Execution on Arbitrum is fully deterministic; i.e., a current chain state along with new input data is sufficient to compute the new chain state; thus, the moment this input data is available (i.e., when the Sequencer posts a batch), the L2 chain's state can be computed.
- Arbitrum's fault-proof system is sound; i.e., if any validator (later) tries to deviate from the valid L2 state, an honest validator will ultimately be able to challenge this and win. Since we already know that valid state will ultimately win out, we can treat our transaction as L1-finalized now.

4. Validator asserts RBlock that includes transaction

A staked, active validator will then run the Arbitrum VM over the inputs in the Inbox (just like the Sequencer did earlier, except now only over transactions posted on L1) and make an on-chain assertion about the chain's latest state, i.e., a rollup block or "RBlock." RBlocks typically get asserted every 30-60 minutes.

See :

- ArbOS
- Geth
- L1 pricing
- /L2 Gas

note RBlock assertions include claims about the state of the Outbox; if our transaction triggered any L2 to L1 messages, a RBlock will include an update to the Outbox to reflect its inclusion. See :

- The Outbox

4a. RBlock is valid / goes unchallenged

In the happy / common case, the validator asserted a valid RBlock, and over the course of the dispute window — 1 week on Arbitrum One — no other validators challenge it.

4b. Assertion is challenged

If two validators assert different RBlocks, only (at most) one of them can be valid, so they are put into a dispute.

A dispute consists of two staked validators dissecting their disagreement down to a single L2 block, and then dissecting the sequence of VM instructions within this block down to a single OPCODE, then finally, executing this single operation. The underlying VM the Arbitrum uses isWebAssembly (Wasm) , or, more precisely, "WAVM." This is all refereed by contracts on L1.

See:

- Challenges
- Wasm/WAVM

L1 contracts also keep track of the tree of all assertions; i.e., how many stakers are in disagreement, who is currently disputing with whom, etc. We refer to this level of Arbitrum's design architecture as its "assertion tree protocol."

See:

- Assertion Tree Protocol

~ ~ ~ FINALITY CHECK: STILL THE SAME Ethereum-Equivalent Finality! ~ ~ ~

Remember in phase 3 when said that once the L1 has committed to inputs, we can guarantee the L2 output? We meant it! Even during a dispute, Arbitrum nodes continue to execute and active validators continue to make assertions on the valid leaf in the state-tree; nothing that can happen in phase 4 has any effect on the L1-level finality we've already locked in at phase 3.

5. RBlock is confirmed on L1

Once any and all disputes have been resolved and sufficient time has passed, our RBlock can be confirmed on L1 (any Ethereum account on L1 can confirm it). Upon confirmation, the Outbox root on L1 gets updated.

~ ~ ~ FINALITY CHECK: L2-to-L1 Messages Executable on L1 ~ ~ ~

If our client's transaction didn't include any L2-to-L1 messages (e.g., withdrawals), phase 5 has no material effect on their transaction. If it did include an L2-to-L1 transaction, it is only after confirmation that the message can be executed in the Outbox on L1.

note Even before phase 5, the client has L1 finality on theresult of their L2-to-L1 message, they just can't execute it yet; i.e., they have a guarantee that they'll eventually be able to, e.g., finalize their withdrawal, they just can't claim their funds on L1 until the RBlock is confirmed. Edit this page Last updatedonApr 29, 2024 Previous Why Nitro? Next The Sequencer and Censorship Resistance