Transaction Lifecycle

In this diagram we observe two domains, "origin" and "destination", representing the chains that a cross-chain message will originate from and travel to.

We also see a differentiation between "fast path" and "slow path".

Fast Path

Requirements

For a cross-chain message to travel through the fast path, it must abide byBOTH of these requirements:

1. The transaction is bridging tokensonly
2. (no calldata) OR the calldata included is unauthenticated (anyone is allowed to call the function on the target contract).
3.

AND

1. Routers are providing sufficient liquidity of the bridged token on the destination domain. We sometimes refer to this as the availability of "fast liquidity". If fast liquidity isnot
2. available, then the message will go through the slow path.
3.

Examples

- A simple token bridge (like theConnext Bridge
-)
- Send funds from origin to destination and then execute a Uniswapswap()
- on destination
-

How it works

Connext is able to shortcut the normal AMB messaging delay (sometimes hours or days of latency!) by allowing its network of routers tofront the capital to the user on the destination domain. Routers wait out the AMB latency in the user's stead, allowing the user to receive funds almost immediately. In exchange for taking on the risk of this temporary liquidity lockup, routers are compensated with a small fee. Note that at the end of the waiting window, routers always get reimbursed.

Slow Path

Requirements

Essentially the inverse of fast path; ifANY of these apply to the cross-chain message, it will travel through the slow path:

1. The transaction includes authenticated calldata (the destination function checks the originating caller from the origin domain).
2.

OR

1. There is insufficient router liquidity of the bridged token on the destination domain.
2.

Examples

- Execute DAO votes across chains
- Change protocol settings from any chain
- Generally, do anything that has anonlyOwner
- modifier or equivalently must validate the origin caller
-

How it works

The message takes on the full AMB delay, allowing the AMB verification process to complete. This is why slow path messages can trust that the origin caller is correct since data integrity is maintained.

Detailed Flow Summary

A transaction flowing through Connext will have the following lifecycle:

- User will initiate the transaction by calling anxcall
- function on the Connext contract, passing in funds, gas details, arbitrary data, and a target address object (includes chain info).
-
    - Note:
-
    - xcall
-
    - is meant to mimic solidity's lower level call as best as possible.
- *
- The Connext contracts will:
-
    - If needed, swap the passed in token to the AMB version of the same asset.
-
    - Call the AMB contracts with a hash of the transaction details to initiate the 60 minute message latency across chains.
-
    - Emit an event with the transaction details.
- *
- Routers observing the origin chain with funds on the destination chain will:
-
    - Simulate the transaction (if this fails, the assumption is that this is a more "expressive" crosschain message that requires authentication and so must go through the AMB: the slow path).
-
    - Prepare a signed transaction object using funds on the receiving chain.
-
    - Post this object (a "bid") to the sequencer.
-
    - Note: if the router does not have enough funds for the transfer, they may also provide only part of the transfer's value.
- *
- The sequencer will be observing all of the underlying chains. Every X blocks, the sequencer will collect bids for transactions. The sequencer will be responsible for selecting the correct router (or routers!) for a given transaction (can be random). The sequencer will post batches of these bids to a relayer network to submit them to chain.
- When a given bid is submitted to chain, the contracts will do the following:
-
    - Check that there are enough funds available for the transaction.
-
    - Swap the router's AMB-flavored funds for the canonical asset of the chain if needed.
-
    - Send the swapped funds to the correct target (if it is a contract, this will also executecalldata
-
    - against the target).
-
    - Hash the router's params and store a mapping of this hash to the router's address in the contract.
-
    -
        - At this point, the user's transaction has already been completed!

- *
- Later, when the slow path message arrives, a heavily batched transaction can be submitted to take all pending hashes received over the AMB and look up whether they have corresponding router addresses in the hash -> router address mapping. If they do, then AMB assets are minted and given to the router.
-
    - Note: if the router gives the incorrect amount of funds to a user or if they execute the wrong calldata, then the router's param hash will not match the hash coming over the AMB and the router will not get reimbursed. This is the core security mechanism that ensures that routers behave correctly.
-
    - Note: Routers will take a 60 minute lockup on their funds when relaying transactions. While this theoretically reduces capital efficiency compared to the existing system, in practice the lack of need to rebalance will mean that routers have more capital available more often regardless.
- *
-

Previous Architecture Next Background Last updated9 months ago On this page *Fast Path * Requirements * Examples * How it works * Slow Path * Requirements * Examples * How it works * Detailed Flow Summary

Edit on GitHub