# Intent Lifecycle in Across

## Intents as a Building Block

Across is used directly by end-users, but it can also be used by other protocols to ensure their user intents are fulfilled. The Across intent structure and lifecycle described below is general enough to serve as a settlement and communication layer for many use cases.

## Initiation

The initiation process involves 3 basic steps: 1. 1. 2. depositV3 3. is called on the 4. SpokePool 5. . This call can be made directly by a user, but could also be called on behalf of a user by some other smart contract system. The chain where this call happens is called the 6. origin chain 7. for this intent. The intent specifies the destination chain, which is where the user wants to receive the output. 8. 2. 9. The user's are pulled into the 10. SpokePool 11. from the caller. These funds are escrowed until the intent is fulfilled, at which point they can be released. Because cross-chain intents are not atomic, user funds must be escrowed before the relayer can safely fulfil the intent. 12. 3. 13. The 14. SpokePool 15. emits the 16. V3FundsDeposited 17. event. Relayers can subscribe to this event to identify intents that they can fill. These steps are intended to be a primitive that can fit into almost any system that requires cross-chain transfers. For instance, this design allows for gasless order systems, where the user simply signs the order and the filler brings it on chain. In such a system, the relayer may be preselected in an offchain auction to minimize the user's cost.

## Fill

After initiation, a relayer must fulfil the user's intent. This process involves three distinct actions: 1. 1. 2. fillV3Relay 3. is called on the 4. SpokePool 5. contract deployed on the destination chain. In this call, the relayer specifies on which chain they would like to receive the user's input tokens. The LP Fee that Across charges on input tokens depends on this choice. Generally, if the relayer takes the input tokens on the chain where the user deposited them, the fee is smallest (if not 0). 6. 2. 7. The intent is marked as filled in the 8. SpokePool 9. . This prevents a second relayer from filling the same intent a second time. 10. 3. 11. The 12. SpokePool 13. emits the 14. FilledV3Relay 15. event. These events can be used to track the status of intents being settled by the system. They are also used to track Note: intents can have an exclusivity period whereby a particular relayer address has the sole right to perform the fill during that period.

## Slow Fill or Expiration (if no fill)

In the (rare) case where a fill doesn't happen, there are two fallbacks: an expiration or a slow fill. A slow fill means that the Across system fills the user without requiring a relayer to provide the capital. It's called a slow fill because it requires Across to optimistically verify this fill before executing it, which means the fill happens a few hours after initiation (much longer than a typical fill). A slow fill happens when the following conditions are met: * The input token and output token are the same asset. * requestV3SlowFill * is called on the destination chain before the expiration time for the intent. * The slow fill is executed before any relayer fills it or the intent expires. In cases where a slow fill can't or does not happen and a relayer does not fill the intent, the intent expires. Like a slow fill, this expiry must be be optimistically verified, which takes a few hours. Once this verification is done, the user is then refunded their money on the origin chain.

## Settlement

Once an intent has been fulfilled, Across verifies that fulfillment and releases the input tokens to the relayer. Across does this by periodically verifying a bundle of intents . The general process for producing, verifying, and executing a bundle is: 1. 1. 2. A block range is determined on each chain supported by Across. This block range goes from the end of the previous bundle's range to a recent block chosen by the proposer. 3. 2. 4. All fill or slow fill request events in the range are validated by ensuring they match some deposit event on the origin chain. 5. 3. 6. All valid fills, slow fill requests, and intent expirations are combined to determine an aggregated list of payments that need to be made on each chain. Those payments are included in the bundle. 7. 4. 8. If funds need to be moved between chains to make these payments, those transfer instructions are included in the bundle. 9. 5. 10. These payments and transfers are organized into a series of data structures called merkle trees whose roots are then proposed on chain to the 11. HubPool 12. along with a bond. 13. 6. 14. Once this proposal passes the challenge period without being disputed, the bundle execution can begin: these roots are sent from the 15. HubPool 16. to each chain's 17. SpokePool 18. via canonical bridges. Funds are also transferred according to the bundle instructions in this step. 19. 7. 20. Once these roots arrive, anyone can execute them to make the payments determined in step 3.

## Intent Structure in Across

An intent in Across is essentially a struct (set of values) that specifies what the user expects to happen. The user's funds are only released once Across verifies that this intent was satisfied as specified. The basic fields are: Field Description recipient the address that should receive the funds inputToken the token that the user supplies inputAmount the amount of the inputToken the user supplies outputToken the token that the user wants to receive outputAmount the amount of the output token the user wants to receive destinationChainId where the user wants to receive the output tokens fillDeadline deadline for the user to receive the tokens message custom data that is sent to the recipient if it's a contract; this allows for custom actions to be executed on the destination chain Advanced fields: Field Description exclusiveRelayer a preselected relayer who is given the exclusive right to fill the user exclusivityDeadline deadline for the exclusive relayer to perform the fill before it is opened to other fillers

Concepts -Previous Intents Architecture in AcrossNext- Concepts Canonical Asset Maximalism

Last modified23d ago