Tracking Events

Implementation

Deposit and corresponding fill events are conveniently scraped by a database and available here . The database implementation can be found in this repository .

How to detect the status of a deposit

When a user deposits capital to a SpokePool, a V3FundsDeposited event is emitted. This event is now emitted for both deposit() (legacy function interface) and depositV3() calls. FundsDeposited , the V2 event, is no longer possible to emit. event

V3FundsDeposited (address inputToken , // Note: outputToken can be set to 0x0 (zero address) in which case, the filler // should replace this address with the "equivalent" destination chain token // as the input token. For example, if input token is USDC on chain A, then the output // token should be the USDC token supported by Across on the destination chain address outputToken , uint256 inputAmount , uint256 outputAmount , uint256

indexed destinationChainId , uint32

indexed depositId , uint32 quoteTimestamp , uint32 fillDeadline , uint32 exclusivityDeadline , address

indexed depositor , address recipient , address exclusiveRelayer , bytes message) This data comprises the new deposit's "RelayData" which is combined with the block.chainId of the SpokePool that emitted the event to form: // This struct represents the data to fully specify a **unique** relay submitted on this chain. // This data is hashed with the chainId() and saved by the SpokePool to prevent collisions and protect against // replay attacks on other chains. If any portion of this data differs, the relay is considered to be // completely distinct. See _getV3RelayHash() below for how the relay hash is derived from the relay data. struct

V3RelayData

{ // The address that made the deposit on the origin chain. address depositor ; // The recipient address on the destination chain. address recipient ; // This is the exclusive relayer who can fill the deposit before the exclusivity deadline. address exclusiveRelayer ; // Token that is deposited on origin chain by depositor. address inputToken ; // Token that is received on destination chain by recipient. address outputToken ; // The amount of input token deposited by depositor. uint256 inputAmount ; // The amount of output token to be received by recipient. uint256 outputAmount ; // Origin chain id. uint256 originChainId ; // The id uniquely identifying this deposit on the origin chain. uint32 depositId ; // The timestamp on the destination chain after which this deposit can no longer be filled. uint32 fillDeadline ; // The timestamp on the destination chain after which any relayer can fill the deposit. uint32 exclusivityDeadline ; // Data that is forwarded to the recipient. bytes message ; } function

_getV3RelayHash (V3RelayData memory relayData)

private

view

returns

(bytes32)

{ return

keccak256 (abi . encode (relayData ,

chainId ())); } There are two ways to determine whether a deposit has been filled on the destinationChain : 1. 1. 2. Each fill of a deposit emits a 3. FilledV3Relay 4. which emits all of the data that you'd need to construct another 5. V3RelayData 6. structure (along with the destination chain's 7. block.chainId 8.). If this relay hash matches the deposit relay hash, then the deposit is considered valid and the filler will receive a refund in the next bundle. 9. event 10. FilledV3Relay 11. (12. address 13. inputToken 14. , 15. // Note: outputToken should never be 0x0 in this event, unlike in a V3FundDeposited 16. // event. If this is 0x0 in the corresponding deposit event, then this should 17. // be equal to the equivalent destination token supported by this SpokePool. 18. address 19. outputToken 20. , 21. uint256 22. inputAmount 23. , 24. uint256 25. outputAmount 26. , 27. uint256 28. repaymentChainId 29. , 30. uint256 31. indexed 32. originChainId 33. , 34. uint32 35. indexed 36. depositId 37. , 38. uint32 39. fillDeadline 40. , 41. uint32 42. exclusivityDeadline 43. , 44. address 45. exclusiveRelayer 46. , 47. address 48. indexed 49. relayer 50. , 51. address 52. depositor 53. , 54. address 55. recipient 56. , 57. bytes 58. message 59. , 60. V3RelayExecutionEventInfo relayExecutionInfo 61.) 62. 2. 63. Call the 64. SpokePool.fillStatuses(bytes32): uint256 65. function passing in the deposit relay hash. The possible fill statuses are: 66. // Fill status tracks on-chain state of deposit, uniquely identified by relayHash. 67. enum 68. FillStatus 69. { 70. // Self-explanatory 71. Unfilled 72. , 73. // Someone requested a slow fill for this deposit, it will be 74. // slow filled to the user in the next root bundle unless it gets filled

75. // by a relayer before that bundle is validated and the slow fill 76. // is executed. 77. RequestedSlowFill 78. , 79. // Filled by a relayer. 80. Filled 81. } Previous Validating Root Bundles Next- Relayers Running a Relayer Last modified23d ago On this page Implementation How to detect the status of a deposit