Selected Contract Functions

Detailed contract interfaces for depositors.

SpokePool state-modifying functions

The spoke pool contract is where deposits are originated and fulfilled and is deployed on all chains that Across supports.

deposit This function is backwards compatible with the existing function with the same name. The main difference in how to use this function is that the relayerFeePct should now be set equal to the LP fee + relayer fee. This is because the concept of the realizedLpFeePct is no longer brought on-chain by the relayer who used to call fillRelay() to fill this deposit. deposit() now emits a V3FundsDeposited event which can only be filled by fillV3Relay() and these fills get charged the LP fee at refund time. Therefore, the caller of deposit() is responsible for setting the relayerFeePct high enough that it covers both the relayer fees and the LP fees when converted into an outputAmount . See Tracking Events to understand more how the new events are emitted by deposit() .

depositV3

This triggers a deposit request of tokens to another chain with the following parameters. The originChainId is automatically set to the chain ID on which the SpokePool is deployed. For example, sending a deposit from the Optimism_SpokePool will set its originChainId equal to 10. Note on sending ETH: deposit is a payable function meaning it is possible to send native ETH instead of wrapped ETH (i.e. WETH). If you choose to send ETH, you must set msg.value equal to amount . Note on receiving ETH: EOA recipients will always receive ETH while contracts will always receive WETH, regardless of whether ETH or WETH is deposited. Note on approvals: the caller must approve the SpokePool to transfer amount of tokens. Note on inputAmount limits: If the inputAmount is set too high, it can take a while for the deposit to be filled depending on available relayer liquidity. If the outputAmount is set too high, it can be unprofitable to relay. Query the suggested max and min limits here . The contracts will not revert if the outputAmount is set outside of the recommended range, so it is highly recommended to set outputAmount within the suggested limits to avoid locking up funds for an unexpected length of time. The recommended outputAmount will be equal to inputAmount * (1 - relayerFeePct - lpFeePct) . Note on setting quoteTimestamp : 1. 1. 2. Call the read-only function 3. getCurrentTime() 4. to get the current UNIX timestamp on the origin chain. e.g. this could return: 1665418548. 5. 2. 6. Call the read-only function 7. depositQuoteTimeBuffer() 8. to get the buffer around the current time that the 9. quoteTimestamp 10. must be set to. e.g. this could return: 600. 11. 3. 12. quoteTimestamp 13. must be <= 14. currentTime + buffer 15. and >= 16. currentTime - buffer 17. . Type Name Explanation address depositor The account credited with the deposit, but not necessarily the one who has to send inputToken s to the contract if the msg.sender differs from this address. address recipient The account receiving funds on the destination chain. Can be an EOA or a contract. If the output token is the wrapped native token for the chain, then the recipient will receive native token if an EOA or wrapped native token if a contract. address inputToken The token pulled from the caller's account and locked into this contract to initiate the deposit. If this is equal to the wrapped native token then the caller can optionally pass in native token as * msg.value, as long as msg.value = inputTokenAmount. address outputToken The token that the relayer will send to the recipient on the destination chain. Must be an ERC20. Note, this can be set to the zero address (0x0) in which case, fillers will replace this with the destination chain equivalent of the input token. uint256 inputAmount The amount of input tokens to pull from the caller's account and lock into this contract. This amount will be sent to the relayer on their repayment chain of choice as a refund following an optimistic challenge window in the HubPool, less a system fee. uint256 outputAmount The amount of output tokens that the relayer will send to the recipient on the destination. uint256 destinationChainId The destination chain identifier. Must be enabled along with the input token as a valid deposit route from this spoke pool or this transaction will revert. address exclusiveRelayer The relayer that will be exclusively allowed to fill this deposit before the exclusivity deadline timestamp. This must be a valid, non-zero address if the exclusivity deadline is greater than the current block.timestamp. If the exclusivity deadline is < currentTime, then this must be address(0), and vice versa if this is address(0). uint32 quoteTimestamp Timestamp of deposit. Used by relayers to compute the LP fee % for the deposit. Must be within depositQuoteTimeBuffer() of the current time. uint32 fillDeadline The deadline for the relayer to fill the deposit. After this destination chain timestamp, the fill will revert on the destination chain. Must be set between [currentTime, currentTime + fillDeadlineBuffer] where currentTime is block.timestamp on this chain or this transaction will revert. uint32 exclusivityDeadline The deadline for the exclusive relayer to fill the deposit. After this destination chain timestamp, anyone can fill this deposit on the destination chain. If exclusiveRelayer is set to address(0), then this also must be set to 0, (and vice versa), otherwise this must be set >= current time. bytes message Data that can be passed to the recipient if it is a contract. If no message is to be sent, set this field to an empty bytes array: ""(i.e. bytes` of length 0, or the "empty string"). See Composable Bridging for examples on how messaging can be used.

speedUpDepositV3

Some of a pending deposit's parameters can be modified by calling this function. If a deposit has been completed already,

this function will not revert but it won't be able to be filled anymore with the updated params. It is the responsibility of the depositor to verify that the deposit has not been fully filled before calling this function.A depositor can request modifications by signing a hash containing the updated details and information uniquely identifying the deposit to relay. This information ensures that this signature cannot be re-used for other deposits.We use the EIP-712 standard for hashing and signing typed data. Specifically, we use the version of the encoding known as "v4", as implemented by the JSON RPC method eth_signedTypedDataV4 in MetaMask.You can see how the message to be signed is reconstructed in Solidity here . Successfully calling this function will emit an event RequestedSpeedUpV3Deposit which can be used by relayers to fill the original deposit with the new parameters. Depositors should assume that the parameters emitted with the lowest updatedOutputAmount will be used, since they are incentivized to use the highest fee possible. (Recall that the relayer's fee is derived by the difference between the inputAmount and the outputAmount). Any relayer can use updated deposit parameters by calling fillV3RelayWithUpdatedDeposit instead of fillV3Relay. Type Name Description address depositor Sender of deposit to be sped up. Does not need to equal msg.sender uint32 depositId UUID of deposit to be sped up uint256 updatedOutputAmount New output amount that depositor requests to receive. Should be lower than outputAmount to be taken seriously by fillers address updatedRecipient New recipient of deposit. bytes updatedMessage Updated data that is sent to updatedRecipient . As described in section above, this should be set to 0x for the forseeable future. bytes depositorSignature Signed message containing contents here

Previous Polygon (Chain ID: 137) Next- Reference Supported Chains

Last modified22d ago