BalanceManager

The BalanceManager shared object holds all balances for different assets. To perform trades, pass a combination of BalanceManager and TradeProof into a pool . TradeProof s are generated in one of two ways, either by the BalanceManager owner directly, or by any TradeCap owner. The owner can generate a TradeProof without the risk of equivocation. The TradeCap owner, because it's an owned object, risks equivocation when generating a TradeProof . Generally, a high frequency trading engine trades as the default owner.

With exception to swaps, all interactions with DeepBook require a BalanceManager as one of its inputs. When orders are matched, funds are transferred to or from the BalanceManager . You can use a single BalanceManager between all pools.

Following are the different public functions that the BalanceManager exposes.

The new() function creates a BalanceManager hot potato (a struct with no abilities). Combine it with share , or else the transaction fails. You can combine the transaction with deposit calls, allowing you to create, deposit, then share the balance manager in one transaction.

The new_with_owner() function creates a BalanceManager hot potato (a struct with no abilities) with a custom owner. Combine it with share , or else the transaction fails. You can combine the transaction with deposit calls, allowing you to create, deposit, then share the balance manager in one transaction.

The owner of a BalanceManager can mint a TradeCap and send it to another address. Upon receipt, that address will have the capability to place orders with this BalanceManager . The address owner cannot deposit or withdraw funds, however. The maximum total number of TradeCap , WithdrawCap , and DepositCap that can be assigned for a BalanceManager is 1000 . If this limit is reached, one or more existing caps must be revoked before minting new ones. You can also use revoke_trade_cap to revoke DepositCap and WithdrawCap .

The owner of a BalanceManager can mint a DepositCap or WithdrawCap and send it to another address. Upon receipt, that address will have the capability to deposit in or withdraw from BalanceManager . The address owner cannot execute trades, however. The maximum total number of TradeCap , WithdrawCap , and DepositCap that can be assigned for a BalanceManager is 1000 . If this limit is reached, one or more existing caps must be revoked before minting new ones.

To call any function that requires a balance check or transfer, the user must provide their BalanceManager as well as a TradeProof . There are two ways to generate a trade proof, one used by the owner and another used by a TradeCap owner.

Only the owner can call this function to deposit funds into the BalanceManager .

Only the owner can call this function to withdraw funds from the BalanceManager .

Only holders of a DepositCap for the BalanceManager can call this function to deposit funds into the BalanceManager .

Only holders of a WithdrawCap for the BalanceManager can call this function to withdraw funds from the BalanceManager .

API

Following are the different public functions that the BalanceManager exposes.

The new() function creates a BalanceManager hot potato (a struct with no abilities). Combine it with share , or else the transaction fails. You can combine the transaction with deposit calls, allowing you to create, deposit, then share the balance manager in one transaction.

The new_with_owner() function creates a BalanceManager hot potato (a struct with no abilities) with a custom owner. Combine it with share , or else the transaction fails. You can combine the transaction with deposit calls, allowing you to create, deposit, then share the balance manager in one transaction.

The owner of a BalanceManager can mint a TradeCap and send it to another address. Upon receipt, that address will have the capability to place orders with this BalanceManager . The address owner cannot deposit or withdraw funds, however. The maximum total number of TradeCap , WithdrawCap , and DepositCap that can be assigned for a BalanceManager is 1000 . If this limit is reached, one or more existing caps must be revoked before minting new ones. You can also use revoke_trade_cap to revoke DepositCap and WithdrawCap .

The owner of a BalanceManager can mint a DepositCap or WithdrawCap and send it to another address. Upon receipt, that address will have the capability to deposit in or withdraw from BalanceManager . The address owner cannot execute trades,

however. The maximum total number of TradeCap , WithdrawCap , and DepositCap that can be assigned for a BalanceManager is 1000 . If this limit is reached, one or more existing caps must be revoked before minting new ones.

To call any function that requires a balance check or transfer, the user must provide their BalanceManager as well as a TradeProof . There are two ways to generate a trade proof, one used by the owner and another used by a TradeCap owner.

Only the owner can call this function to deposit funds into the BalanceManager .

Only the owner can call this function to withdraw funds from the BalanceManager .

Only holders of a DepositCap for the BalanceManager can call this function to deposit funds into the BalanceManager .

Only holders of a WithdrawCap for the BalanceManager can call this function to withdraw funds from the BalanceManager .