Swaps

DeepBook provides a swap-like interface commonly seen in automatic market makers (AMMs). Unlike the order functions, you can call swap_exact_amount without a BalanceManager . You call it directly with Coin objects instead. When swapping from base to quote, base_in must have a positive value while quote_in must be zero. When swapping from quote to base, quote_in must be positive and base_in zero. Some deep_in amount is required to pay for trading fees. You can overestimate this amount, as the unused DEEP tokens are returned at the end of the call.

You can use the get_amount_out endpoint to simulate a swap. The function returns the exact amount of DEEP tokens that the swap requires.

Following are the endpoints that the Pool exposes for swaps.

Swap exact base quantity without needing a balance_manager . DEEP quantity can be overestimated. Returns three Coin objects:

Some base quantity may be left over, if the input quantity is not divisible by lot size.

You can overestimate the amount of DEEP required. The remaining balance is returned.

Swap exact quote quantity without needing a balance_manager . You can overestimate DEEP quantity. Returns three Coin objects:

Some quote quantity might be left over if the input quantity is not divisible by lot size.

This function is what the previous two functions call with coin::zero() set for the third coin. Users can call this directly for base → quote or quote → base as long as base or quote have a zero value.

API

Following are the endpoints that the Pool exposes for swaps.

Swap exact base quantity without needing a balance_manager . DEEP quantity can be overestimated. Returns three Coin objects:

Some base quantity may be left over, if the input quantity is not divisible by lot size.

You can overestimate the amount of DEEP required. The remaining balance is returned.

Swap exact quote quantity without needing a balance_manager . You can overestimate DEEP quantity. Returns three Coin objects:

Some quote quantity might be left over if the input quantity is not divisible by lot size.

This function is what the previous two functions call with coin::zero() set for the third coin. Users can call this directly for base → quote or quote → base as long as base or quote have a zero value.