System Overview

The `mmvault` contract is a **liquidity management vault** deployed on the Neutron blockchain. Its primary purpose is to manage user deposits of two tokens (e.g., `token0` and `token1`) and efficiently allocate liquidity into the Neutron DEX (Duality) by performing automated deposits and withdrawals, driven by off-chain price data and strategic configuration parameters.

The vault contract abstracts away direct DEX interactions from end users. Instead, users interact with the vault through standardized deposit and withdrawal flows, while the vault itself takes full custody of user assets and autonomously manages liquidity provisioning via Cron-triggered rebalancing logic.

Core Flows

User Deposit

Users can deposit paired tokens into the vault using the `ExecuteMsg::Deposit` entry point. In return, they receive **LP shares** (minted using the token factory module) representing their proportional claim on the vault's total value. Shares are minted based on:

- Vault's total value (fetched using slinky price oracle),
- Existing LP supply (`config.total_shares`), and
- Whether the deposit is the first-ever or a subsequent one.

Shares are minted using the `get_mint_amount()` logic, ensuring proportional issuance. LP share tracking is maintained through `config.total_shares`, which increases on every deposit.

User Withdraw

Users can redeem their LP shares via `ExecuteMsg::Withdraw`, triggering a burn of the specified amount of LP shares. If the vault holds no active DEX positions, the withdrawal immediately transfers the user's share of vault-held tokens using `get_withdrawal_messages()`. If active DEX deposits exist, the vault issues a submessage to withdraw them (`MsgWithdrawal`), and the actual token payout happens in the reply handler, followed by a fresh `MsgDeposit` to redeploy remaining liquidity.

The withdrawal flow ensures that `config.total_shares` decreases by the exact burned amount.

Automated DEX Rebalancing via Cron module

The vault relies on Neutron's **Cron module** to trigger periodic rebalancing of its DEX positions through two privileged entry points:

1. `dex_withdrawal()`:

   Executed first each block by Cron, this function:

   - Queries all active DEX positions owned by the vault,
   - Issues `MsgWithdrawal` messages to remove liquidity from all active pools,
   - Ensures the vault regains custody of all tokens before re-depositing.

2. `dex_deposit()`:

   Executed immediately after `dex_withdrawal()`, it:

   - Verifies that all prior positions have been withdrawn,
   - Fetches fresh oracle prices via the **Slinky module**,
   - Computes the target tick index and rebalancing logic based on:
     - Fee tier allocation,
     - Imbalance between token0/token1 values,
     - Configured skew/imbalance settings,
   - Issues `MsgDeposit` messages to redeploy liquidity across different price ranges and fee tiers.

All deposits and withdrawals are issued on behalf of the vault (`creator == receiver == contract address`), ensuring it retains full custody of all funds and LP shares.

Appendix: Diagrams

All main execution flows—including user `deposit`/`withdraw` interactions, cron-triggered DEX operations, as well as state machine transitions are illustrated in the **Appendix:**

Diagrams section of this report.