Mechanisms of action

Terminology

- `pool` A contract with two pairs that tracks changes in an asset over a set period. Denominated in a digital asset.
- `pair` One half of a pool. Each pool has a short and a long pair. Pair balances of the digital asset are moved during a price execution.
- `update interval` The time in seconds between the execution of a price change in a pool.
- `frontrunning interval` The length of time in seconds before a price update execution that a user must commit for the commitment to be executable in the following update interval.
- `price execution` A movement of funds from one pair to the other. The direction will depend on the price change
- `shadow pool` The combined value of all unexecuted commits of a certain type. There are 4 types of shadow pool (Long mint, long burn, short mint, and short burn). The mint types represent amounts of the digital asset to be added to the pool. The burn types represent claims on the digital assets of a given pool.
- `live pool` The balance each pair holds in the digital asset. There are only two live pools. These funds are what is moved during a price execution.
- `market data` Price data for a specific market code and update interval pair.
- `upkeep` The combination of update interval, market code, and one or more pools. It is possible (but not required) to register an upkeep with the Chainlink keeper network.

Depositing into a pair (Minting)

The process for depositing into a pair is identical for both pairs. A user can execute multiple commits at a time. These are processed in order as described here and in `Withdrawing from a pair (Burning)`

1. The user must commit to depositing. This will store a record of their commit, accessible by ID. The user must retain a record of this id so they can execute it later.
2. The user's commit is added to the shadow pool for the commit type.
    - Creating a commit will transfer the requested amount of the digital asset from the user's wallet into the contract.
3. After the next price execution occurs, the user can execute their commit.
    - For this to be possible, the user must have added the commit between the start of the interval, and the interval length minus the frontrunning interval.
        - If the frontrunning interval is 5 minutes, and the next price execution is at 20:00, then the user must commit before 19:55 for the commit to be executable immediately after the price execution.
    - This is enforced by checking `last execution time - commit creation time > front running interval` when executing a commit.
    - If the user did not commit before the frontrunning interval, they must wait for the next interval to execute their commit.

4. Upon execution, the following happens:
    1. The commit amount is moved from the shadow pool into the relevant live pool for the commit.
        - For instance, executing a commit to deposit into the short pair will add the commit amount to the live short balance and subtract it from the shadow short mint balance.
    2. An amount of pair tokens is minted and assigned to the user's wallet address.
        1. The exact amount is determined by `Amount of the digital asset committed * (Total number of pair tokens issued / Total amount of the digital asset held by the pair)`.

Withdrawing from a pair (Burning)

The process is the same for withdrawing from either pair.

1. The user must have a balance of the pair token that they wish to redeem.
2. The user creates a commitment to withdraw.
    1. If this occurs within the front running interval for the pool, they must wait until the interval after the next one before the commitment can be executed. The process for this is described in `Depositing into a pair (Minting)`
    2. Creating a commitment will burn the user's committed amount of pair tokens. This is done to remove the requirement for transfer approval before committing.
3. After the required delay, the user can execute the commitment to withdraw.
    1. This will transfer an amount of the digital asset to the user.
    2. The amount of the pair's digital asset that the user receives is given by `Number of pair tokens committed * (Total amount of digital asset held by the pair / total number of pair tokens issued)`. When performing this, the balance of the tokens that were burned is taken into account.
    3. The shadow pool balance for the burn type will be decreased by the amount that was committed.

Withdrawing a commit

The user can withdraw any commit provided that it hasn't already been executed. The user must own the commit that they are withdrawing. There are no restrictions on when a user can withdraw a commit.

For mint commits:

1. The digital asset is returned to the user
2. The balance that was returned is subtracted from the shadow pool
3. The commit is deleted

For burn commits:

1. The amount of pair tokens committed are minted to the user's account (They were burned on commit)

2. The amount of pair tokens being refunded is subtracted from the shadow pool
3. The commit is deleted

Creating a market and a pool

The ability to create a market or a pool within a market is open to anyone.

Creating a market will add a record for the market code and oracle to the oracle wrapper. The market details will be emitted in an event.

Creating a pool will:

1. Take the current price for the market
2. If this is the first pool to use the market data:
    1. Start a new interval
    2. Add a price sample to the market/update interval price data
3. Otherwise, if the current price and the last price sample taken are not the same, a new sample will be added to the price data for the market/update interval
4. A new pool will be deployed using the pool factory. Details will be emitted via event.

Pool upkeep

The pool implements the Chainlink keeper interface for upkeep. There are two types of upkeep: market data management, and pool maintenance.

An upkeep is made up of a market, an update interval, and one or more pools. New upkeeps are registered in the pool keeper the first time an unkeep is performed. The pool keeper will track the following to ensure pool security:

- The last execution time of a pool
- The market that a pool belongs to
- The pool's update interval

Market price data is stored under pairs of market code and update interval. Multiple pools at different leverage points can use the same price data, as long as they all share the same update interval and market.

An upkeep will occur when one of the following conditions is met:

1. The price for the market and update interval has changed from the last sample taken, or
2. The last execution time for one or more of the pools in the upkeep is less than the start time of the current update interval (this occurs when a new interval is started)

Market data management (occurs during an interval when the oracle price changes):

1. A price sample for the current update interval is taken
2. Optionally a price update execution will occur. This will update each pool in the upkeep with the new price, causing them to adjust the live balances accordingly. This will only happen if there is more than one upkeep for a given market/update interval pair. It is possible for multiple

upkeeps to use the same market data.

Pool maintenance (occurs when an interval has ended):

1. Start a new interval. The average price for the interval will be calculated and stored.
2. Execute a price update for each pool in the upkeep using the new average price, and the previous average price.

Events

To reduce the amount of data stored on chain, and save on gas, events are used as a form of storage. In particular, commits, and pool addresses are not accessible without knowing their identifier (commit id, and pool code, respectively). Frontend clients should be aware of this, and save event logs accordingly.