The OADA System

ZYGOMEB
Optim Finance
zygomeb@gmail.com

Document Version: 1.0
System Version: 1.0
2024-05-19

For the latest version visit [THIS REPOSITORY] .

PRE-DEPLOY VERSION, FOR THE PURPOSES OF AIDING THE BUG BOUNTY PROGRAM

SOME FACTS FORWARD REFERENCE STATE OF REALITY AT LAUNCH NOT PRESENT

The document captures the current state of the deployment of the system, sufficient reasoning and explanation of the system operations, as well as the changelog from past versions.

It is presented as-is for purely informational purposes.

A system labeled as an Algorithmic Market Operation (AMO) is a Strategy equipped with the power to manipulate the supply of OADA by minting and burning using associated rules.

1. High Level Description

Shadow looms over the ghost of an ethereal beauty, a raw force of financial warfare; form of a spinless cutthroat efficiency, taking root between the cracks of those that take their power for granted. Harnessed, for now. Made tame, able to be understood. Made humane.

OADA System is an instantiation of the OTOKEN framework to create a fully-backed ADA derivative token. It is currently range-pegged within one percent of 1:1, and rebalances the open market price of OADA to keep the price of the asset within that range via buybacks and arbitrage.

Autocompounding and able to take advantage of ecosystem yield opportunities, the system passes their fruits on to the staked version – that is, both taking on the risk of the strategies as well as the yield from them. It is intended and structured to be as low risk as possible, limited in intent to only smart contract and similar low level lending risks.

Allocation of system reserves and invocation of the majority of AMOs requires a signature of one of the system operators referred to as a Controller. It is a governance-delegated position, that possesses no custody of the assets, in all situations aside from a potential system exploitation.

The OADA system is governed by the ODAO, via the use of a Soul Token that can upgrade it in any way necessary.

2. Status Report

This section we will contain high level milestones, as well as highlight points that show the system's history.

Currently the system is about to be deployed, as described by system version 1.0.

3. Relevant Information

All of the information in this section contains important context, extrinsic to the system

The ODAO treasury has deposited TODO ADA and is keeping it unstaked to boost the initial adoption of the system, as well as providing TODO paired with equal amount of ADA to keep a long term liquidity base for the system's operations.

OADA Soul Token is currently held in a multisignature wallet between Optim Labs members and ODAO Council members. It is intended to be transferred over to a fully decentralized governance system once one is built up, fit for the protocol's requirements.

4. System Solvency Analysis and Risks

When it comes to the risk profile of the system, for actions that mint OADA (rules) we aim to ensure the highest degree of safety of the system. Here we compile the risks for counterparties of the OADA system and their risks, for every strategy-type AMO, as well as arguments for what makes them safe to execute.

Further elaboration on each strategy is given in their own sections, here only outlining the key risks of them.

4.1. DEX AMO Risks

- Currently only interfacing with the Splash OADA/ADA stableswap pool
- The Splash system is quite new, and without much capital in it and therefore not battle-tested
- Splash DAO governance can tweak fee parameters on the pool and potentially cause the system principal loss
- The Stableswap pool is open source and audited

Strategy Added Risks

4.1.1. Peg Protection Swaps

The system guards against making a transaction that is unprofitable. Meaning, it is impossible for the system to lose money through swapping between OADA and ADA.

4.1.2. Liquidity Provision

It can be accurately described, from another perspective, as a leveraged (~2x) LP strategy with no cost of leverage.

We use the commonly known fact that the sum of tokens deposited does not decrease (and in fact, increases) away from the center point of the curve for any amplification parameter of the canonical stableswap.

This implies that, for deposits that are made at a price point closer to a 1, if the price goes further away from the 1, the sum of assets backing the LP increases. An important feature for OADA as we can take advantage of the symmetry of the stableswap curve in combination with system accounting for any OADA profits as = 1 ADA (and conversly), to assert that by depositing liquidity in the 1:1 center of the AMM, can never lose money via rebalancing loss (also can be called, the no impermanent loss property).

We want to, however, to be able to deposit *not at the perfect center*, so that only a minimum amount of reserves are used to rebalance the pool prior. The mathematical proof doesn't hold for such deposits, and we're forced to refer to our numerical analysis that shows that with a swap fee of 0.1% is sufficient to guarantee that no rebalance can cause the final withdrawn sum to go under the initial sum of deposited assets.

Furthermore, in presence of a swap fee, it may be impossible to rebalance the pool to the perfect center without taking a loss on the rabalancing swap. This analysis was done for 100, 150 and 200 amplification argument stableswap pools.

A constraint on deposits is that the pool doesn't sell 1 OADA for less than 1 ADA, so we are limited to deposits in range of prices above 1-swapfee. The final safe deposit range for a 0.1% fee pool going to the LP is the price range of 0.999-1.001 ADA per OADA.

The above range is the mathematically perfect range. Some error between a nonzero splash protocol fee and a fractional number error safety buffer gives us a slightly tighter range. Which implies, also, that with the current restrictions on arbitrage swaps (no loss on swap), there exists a very small price range where the protocol is neither able to deposit, nor bring it to a price point at which it will be able to.

Such states are not realistically profitable for any potential attacker to exploit, nor is it realistically possible to keep an open market price perfectly within the very tight (0.05%) TODO inactive range. Even if such an attacker is assumed to be Byzantine, the system is under no risk if this part of the DEX AMO strategy is not able to execute.

It goes without saying that the withdrawal counterpart of the equation does not have any such limits.

4.2. Staking Auction AMO Risks

- This is a strategy providing liquidity to a market made as a dapp internal to the OADA system considered part of the AMO as a whole

Strategy Added Risks

- Gradually locks up system ADA reserves during the epoch, and provided users are willing to pay a high price, they can acquire all of the reserves of the system.
  - The pricing curve gets increasingly steep, upwards of paying double and even quadruple digit APY for a single epoch in order to acquire all of the reserves early in the epoch
  - It is still, however, possible to acquire. The system is very much prepared to accept any such offers.
- Funds locked up aren't able to be used in other strategies
  - Noteworthily, also not able to be used to protect the peg of the OADA/ADA pair. As such, it can be expected that at such times, without users willing to acquire it under peg, open market price will be totally steered by speculation.
  - Upon the start of the new epoch, the assets are released, and able to be used to perform any function, including a repegging of the pair
  - Oracles during such market conditions can and will be affected and systems not prepared to deal with such system operations should not be used to acquire leverage on assets trading against OADA or against OADA or sOADA themselves.

4.3. Staking AMO Risks

- Unlike the other modules in this section, this is a note about risks associated with using the Staking AMO (staking/unstaking OADA for/to sOADA)
- The main risk associated with staking OADA is that in the case of a system loss – it bears the cost of the loss. It can be thought of as the Junior tranche, that gets all the yield bearing the risk of covering for system loss.
- System loss under normal operations should not occur as the system takes on a minimized risk exposure.

5. Capital Flows

In this section we break down the system's operations as performed by the currently delegated Optim Labs Controller.

5.1. Schedule and Operations

For OADA to run smoothly and transparently we must have well-defined semantics for each of its actions, as well as perform them. As a simple mental model of how the capital flows through the system, the ADA gets deployed into strategies during the epoch and at the end of it, the staking rights are sold to the highest bidder. After the epoch boundary, all of that ADA gets redeployed back into the strategies.

5.1.1. Deposit-Deploy

As users mint OADA with ADA, that ADA sits at the Deposit Address waiting to be processed. Processing happens either at the end of the epoch, or once enough of it minted. This is done in order to have the controller save on transaction fees without losing any meaningful amount of profit for the system.

The current threshold for the Controller to process deposits is 100 000 ADA in total pending.

When a Deploy routine is triggered, whether by a deposit threshold or by beginning-of-epoch redeployment, the system looks up the current distribution of assets and distributes the inflow of capital targeting up to a 2% deviation from the 'ideal' set distribution between strategies. This is purely a Controller guideline, and it has the freedom to act outside of it and distribute it as it believes is most beneficial.

Currently the distribution target is 100% to the DEX AMO, as there are no other strategies deployed.

5.1.1.1. Lending AMO Deploy

The only lending market configured is the Liqwid ADA market, which upon receiving new capital, it simply adds it to the pile of ADA that's already being used in the pooled lending protocol.

5.1.1.2. DEX AMO Deploy

DEX AMO operates by keeping, for the sake of efficiency, a small part of the total funds (10%) it controls idle, and waiting to be used for peg keeping by OADA buybacks. This number will be revised in the future to ensure high degree of efficiency.

If the AMO controls 0 ADA right now (first deploy of the epoch) then a purchase of OADA is made to bring it up to 0.999 ADA per OADA price. This amount of capital is pre-calculated during distribution between strategies and isn't included in the 30% the DEX AMO gets, but rather the 30% is calculated after this amount is subtracted from the total reserves.

What the purchase allows, is a safe deployment of ADA paired with minted OADA in the exact same ratio as the DEX pool. Assuming that the swap fees are 0.1% per swap or above, this makes depositing of minted OADA in this ratio to ADA deposited an action that cannot lower the backing of it from 1:1, as any ADA entering a swap will not cause a leak of OADA acquired for less than 1 ADA unless someone sells it for less than that first, which balances itself out entirely.

5.1.2. Clearing-Rate-Match Event

The Stake Auction of OADA is done continually all throughout the epoch until it is time to Undeploy-Auction the rest of the system reserves – this way, with a Continuous Dutch Auction, we are able to minimize the ability for bidders to strategically place orders and ensure the highest profitability for the system given demand for stake.

To determine the current price of stake, the controller works with a pricing curve based on the time passed in epoch, and remaining reserves to be sold – full equation in its section. It uses a minimum match size of 250k ADA stake and matches bids on a FCFS basis sorted first by the APY offered.

When enough bids to get a minimum size match are found, such that the system, after matching them, will reprice clearing rate to be at least as much as it has just received – the system will match these orders and lock the stake with their chosen stake key until the epoch ends.

5.1.3. Undeploy-Auction

Before the epoch ends, an undeploy event takes place, which pulls all available assets out of the system. This is also the time when the system guarantees profits to be synced and pushed to affect the sOADA/ADA exchange ratio.

The ADA is then moved to fill stake bids (which were placed and filled all throughout the epoch) in the Stake Auction AMO, where it resides up until the end of the epoch, and afterwards syncs profits from the auction and moves to deploy the funds in the rest of the system. In other words, settlement of the actual auction takes place within one hour before the epoch ends.

5.1.4. Pegkeeper-Capitalize Event

Unlike the purchase of OADA during the deployment at the start of the epoch, during the epoch the system only protects the peg when it deviates under and up to 1% down from the 1:1 status.

5.1.5. Voltaire Notes

The system allows Voltaire vote power to be acquired, as it is derived from one's stake key weight. Given how the governance process of Voltaire works, everyone will know 1 epoch ahead of time if the next epoch has any important votes going on. Equivalently, there can be no last minute surprises to ratify a proposal without it going through at least 2 epoch boundaries first (1 boundary to allow voting, and the next boundary to see if it ratified or otherwise).

5.1.6. Catalyst Snapshot Event

If a Catalyst vote power snapshot is to take place, and it is taken close to an epoch boundary, the system has to decide whether to auction off the stake power OR keep it for the ODAO. An example (as currently it is not yet decided) policy is to only sell to bids offering 10% APY or higher for that epoch boundary. If it is not close to a boundary, then the system will simply capture all of the vote power for the ODAO.

When vote power is captured by the ODAO, the ADA rewards for voting will be passed into the OADA system using the Donation Strategy, including the stake rewards, if on an epoch boundary.

Note: Given the centralized nature of the snapshot (especially when no exact snapshot slot is given) the Controller cannot guarantee that this operation will be carried out as described here.

6. Active Modules

The active modules, for the sake of categorization and convenience are grouped into Packages. Currently there are two packages, the Core and the Base packages.

Within Core is the minimal version of the system containing all of the accounting operations logic. It is audited by Anastasia Labs and fully open sourced

Base Package contains the necessary AMOs for the system's operations including interoperation with closed source systems like Liqwid. It is neither audited nor open sourced, as doing so requires all of the systems it integrates with to open source first for inspection of the auditors. In case of any problems found in a potential future audit, once possible, the system can gracefully upgrade them via governance.

6.1. OADA Token

Introduced in	1.0
Last Revision	1.0
Package	Core
Source	https://github.com/OptimFinance/clean-code
Audits	https://github.com/OptimFinance/clean-code
Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH
OADA Token is an extensible set of rules for minting and burning OADA. It is governed by a soul token and intrinsically has no ties with the rest of the system beyond what rules define. One could, in theory define additional rules that interact with components completely external to 'the system' as we speak here – it is easily observed that this method of defining a token makes it possible to define a completely asynchronic and everpresent asset with rules that span the entire market.	

6.1.1. OADA Rule Whitelist

Introduced in	1.0
Last Revision	1.0
Package	Core
Source	https://github.com/OptimFinance/clean-code
Audits	https://github.com/OptimFinance/clean-code
Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH
First mention of this pattern in this document, additional elaboration of the pattern is given. The whitelist is a set of fractioned data that is used to authenticate and prove witness to a governance process having been processed successfully, that is, a witness of a soul token was present upon the creation of a whitelist record as a mint of a whitelist NFT is made only possible by that.	

Here we use this pattern to add ‘Rules’, aforementioned fractional invocations that define when it is possible to mint and burn OADA. It is done via the OADA policy looking for an authenticated self NFT-holding reference utxo that has a datum with a script that is being invoked as a withdrawal. And if that is present, anything is permitted.

6.1.2. OADA Whitelisted Rules

In this section we record the currently deployed rules for minting/burning OADA. They are explained under the AMO that is rellated to them, to give necessary context to their functionality.

6.1.2.1. OADA Deposit Rule

6.1.2.2. OADA/sOADA Staking Rule

6.1.2.3. OADA Fee Claim Rule

6.1.2.4. OADA Splash DEX Rule

6.1.2.5. OADA Stake Auction Rule

6.2. sOADA Token

Introduced in	1.0
Last Revision	1.0
Package	Core
Source	https://github.com/OptimFinance/clean-code
Audits	https://github.com/OptimFinance/clean-code
Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH

In its technical design, sOADA is equivalent to OADA. It is only the presence of different rules that gives the semantical meaning to the token.

6.2.1. sOADA Rule Whitelist

Introduced in	1.0
Last Revision	1.0
Package	Core
Source	https://github.com/OptimFinance/clean-code
Audits	https://github.com/OptimFinance/clean-code

Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH
As above.	

6.2.2. OADA Whitelisted Rules

In this section we record the currently deployed rules for minting/burning sOADA. They are explained under the AMO that is related to them, to give necessary context to their functionality.

6.2.2.1. OADA/sOADA Staking Rule

6.3. Deposit AMO

Introduced in	1.0
Last Revision	1.0
Package	Core
Source	https://github.com/OptimFinance/clean-code
Audits	https://github.com/OptimFinance/clean-code
Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH
The Deposit AMO is the simplest possible system, that also illustrates the bottom line of the economic system model. It allows for 1 ADA that enters the AMO (and into the system), 1 OADA to be minted. As a general rule of thumb, this is the simplest and most robust way to ensure a peg can be kept. It can, however, be expanded to the notion of overcollateralized, allowing it to be issued against collateral instead. This, as seen by other protocols such as Maker, is quite a profitable business. It must, however, be done with caution as overissuance of tokens in this way without ensuring the stability of the asset will hold price on the open market will lead to price target (peg) failure. Currently the system has no overcollateralized avenues to issue the token.	

6.3.1. OADA Deposit Rule

Introduced in	1.0
Last Revision	1.0

Package	Core
Source	https://github.com/OptimFinance/clean-code
Audits	https://github.com/OptimFinance/clean-code
Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH
Deposit Rule simply allows minting of X OADA if X ADA is sent to the Deposit AMO. It has a minimum of 100 OADA minted to prevent dusting attacks.	

6.4. Staking AMO

Introduced in	1.0
Last Revision	1.0
Package	Core
Source	https://github.com/OptimFinance/clean-code
Audits	https://github.com/OptimFinance/clean-code
Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH
All of the accounting within the contracts is headed down a pipeline from the strategies through the CM AMO to the Staking AMO. It tracks and accounts for the DAO Fee, and distribution of it to the sOADA holders by way of changing the sOADA/OADA exchange rate. Of importance are the **odao_fee** and **staking_cap** parameters that can be changed by governance, and dictate a % cut of total system profit routed towards the DAO treasury and the maximum amount of sOTOKENS. The ODAO fees accumulate in sOADA and circumvent the cap on staked amount upon accrual, and may be claimed anytime by the fee claimer token, the only governance witness token of the system other than the soul token itself.	

6.4.1. Batch Stake Request

Introduced in	1.0
Last Revision	1.0

Package	Core
Source	https://github.com/OptimFinance/clean-code
Audits	https://github.com/OptimFinance/clean-code
Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH
Using a direct method of accessing the Staking AMO utxo was originally planned, but for reasons of accessibility and spam prevention – as well as enabling the creation of a stake queue, we introduce a necessary intermediation script that is to be processed by the Controller in a FIFO manner.	

6.4.2. OADA/sOADA Staking Rule

Introduced in	1.0
Last Revision	1.0
Package	Core
Source	https://github.com/OptimFinance/clean-code
Audits	https://github.com/OptimFinance/clean-code
Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH
The rule is used both by sOADA and OADA, and guards the staking and unstaking process. Specifically, it ensures that mints of sOADA happen only when OADA is burned in a transaction that has the Staking AMO, and that the mint is done at the current sOADA/OADA exchange rate, as well as that it was indeed recorded in the total staked amount. The opposite way, to unstake/burn sOADA and mint OADA, the same process takes place, with an unstake fee of **0.1%** of the total.	

6.5. Collateral Management AMO

Introduced in	1.0
Last Revision	1.0

Package	Core
Source	https://github.com/OptimFinance/clean-code
Audits	https://github.com/OptimFinance/clean-code
Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH
The Collateral Management AMO is at the heart of the system, from it all the AMOs flow and report to, accounting of profits and potential losses; the CM AMO is recording all the created instances of whitelisted strategies to allow them free capital flow and accounting through all of the system's tendrils.	

6.5.1. NFT ID Policy

Introduced in	1.0
Last Revision	1.0
Package	Core
Source	https://github.com/OptimFinance/clean-code
Audits	https://github.com/OptimFinance/clean-code
Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH
A utility token that allows the creation of a provable globally unique non fungible asset. Used to identify strategies by their IDs, and most importantly, the CM AMO by its ID that in turn defines the entire system.	

6.5.2. Controller Whitelist

Introduced in	1.0
Last Revision	1.0
Package	Core
Source	https://github.com/OptimFinance/clean-code

Audits	https://github.com/OptimFinance/clean-code
Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH
A whitelist, governed by ODAO, that holds the PubKeyHashes of Controllers. For many of the system transactions, a signature by one such actor is required. Context as to why they must be guarded is simple: a decision problem and guarantee of a schedule (as well a definition of such schedule) would need to otherwise be put onchain to guard against malicious actions that can't lose the system principal but can otherwise make the system behave erratically and cause disruption in the markets.	

6.5.3. Strategy Whitelist

Introduced in	1.0
Last Revision	1.0
Package	Core
Source	https://github.com/OptimFinance/clean-code
Audits	https://github.com/OptimFinance/clean-code
Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH
A whitelist, governed by ODAO, that holds ScriptHashes of strategies that can be created.	

6.6. Stake Auction AMO

Introduced in	1.0
Last Revision	1.0
Package	Base
Source	Not Open Sourced
Audits	None Currently
Plutus Version	2

Unparameterized Script Hash	HASH
Deployed Script Hash	HASH

Stake Auction AMO is the second most important AMO of the system. It handles all of the system's staking operations operating in four steps:

1. Throughout the epoch anyone can create a bid, an offer of ADA or OADA payment for the system's staking power – For a single epoch snapshot.
   - If bids made totalling at least 250k ADA crosses the Clearing Rate, it will be matched then and there, without waiting until the epoch's end.
2. One hour before the epoch ends, no bids can be canceled or made. The system undeploys ADA out of other strategies and deploys it into the Staking Auction AMO to fill bids from the highest APY offered to the lowest until it can match no more. Bids made with OADA are treated as being 0.2% higher and the OADA is immediately burned upon being matched.
3. Bids filled in the previous step sit in a Lock, staked with the bid owner's key, until the next epoch one hours later. They are then unlocked and all of the ADA gathered.
4. The ADA is redeployed into the other strategies for the duration of the epoch.

The System's Clearing Rate is defined as the following function:

$$\mathrm{BR} + \mathrm{PR} * \left(1 - \frac{1}{1 + \exp\left(-\frac{T-50}{100} * 2\right)}\right) * \begin{cases} \left(1 + \frac{\frac{\mathrm{SRIL}}{\mathrm{SR}} - \frac{T}{100}}{1 - \frac{T}{100}}\right)^{5.6 + 2.6 * \left(\frac{T}{100}\right)} & \text{if } \frac{T}{100} < \frac{\mathrm{SRIL}}{\mathrm{SR}} \\ 1 \text{ otherwise} \end{cases}$$

Where
- SR = System Reserves (in ADA, total)
- SRIL = System Reserves In Locks (already offloaded stake)
- PR = 2% = Projected Rate (What the system expects the opportunity cost of not matching bids to be)
- BR = 2.7% = Base Rate (What is the bid floor, set close to the staking rate on ADA)
- T = Time in epoch, a number from 0 to 100 uniformly dividing the entire cardano epoch

The Clearing Rate is enforced by the Controller, and can be changed without updating the smart contract and will be updated by it with oversight and approval of the ODAO Council based on the current state of the market and analysis performed.

It's worth noting that the formula is subject to the dynamic inflow and outflow of reserves from and out of the system.

6.6.1. Stake Auction Bid

Introduced in	1.0
Last Revision	1.0
Package	Base
Source	Not Open Sourced
Audits	None Currently

Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH
The Bid is a simple script that enables the user-facing functionality of the Stake Auction AMO. It has an owner that can cancel it anytime except in the last 1 hour of the epoch. And it can be filled by a Stake Auction AMO, ensuring that the owner receives the stake power.	

6.6.2. Stake Auction Lock

Introduced in	1.0
Last Revision	1.0
Package	Base
Source	Not Open Sourced
Audits	None Currently
Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH
An even simpler script that tracks the epoch it was locked in, allowing unlock to the Stake Auction AMO that matched the bid in the next epoch.	

6.6.3. OADA Stake Auction Rule

Introduced in	1.0
Last Revision	1.0
Package	Base
Source	Not Open Sourced
Audits	None Currently
Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH

A yet even simpler simpler script that allows OADA received from bids into the Stake Auction AMO to be burned.

6.7. Splash DEX AMO

Introduced in	1.0
Last Revision	1.0
Package	Base
Source	Not Open Sourced
Audits	None Currently
Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH

Quite possibly the most complex script in the system. A good joke here would be to not elaborate at all, but – What the system can do is broken up in 4 actions.

1. Protect Bottom Peg
   - The system can buy back OADA as long as the amount bought back is >= amount spent.
   - It needs liquidity sidelined in order to function. If the system runs out of idle capital, it either removes part of the supplied liquidity, or pulls it out of other strategies.
   - This functionality is invoked in two cases
     1. Protecting the Bottom Peg
     2. Rebalancing the pool for the addition of liquidity

     In the first case the controller chooses to protect the downside of the asset lower than the parameter may lead to believe alone, as discussed in an earlier section.

     Currently via the first action the pool gets rebalanced up to price of **0.99** ADA per OADA, and via the second, **0.999**. The second of those two numbers is not 1:1 because of an implied 0.1% swap fee.

     The difference between OADA received (and burnt) and ADA sent is profit for the system.
2. Protect Upper Peg
   - The system can sell freshly minted OADA as long as the amount of ADA coming in >= amount minted. The difference between those two is the system profit. This can be thought of as a round-about way to mint OADA, equivalent to using the Deposit AMO
   - Does not have any need for idle liquidity, and works by invoking the DEX Rule.
3. Add Liquidity
   - Uses reserve ADA paired with freely minted OADA via the DEX Rule to supply liquidity.
   - Can only perform the action (and mint the OADA to pair) if the price on the open market is at least **0.999** ADA per OADA, and no more than **1.001** ADA per OADA. This, counting in a 0.1% swap fee, ensures that none of this minted OADA can be issued (leave the LP) for

a price of less than 1 ADA – keeping the token 1:1 backed. For a more formal proof of system property, refer to a later section.
  - The above assurance holds regardless of swaps, deposits and withdrawals into and out of the pool after the system supplied the liquidity, as trivially seen.
  - Records the sum of the OADA and ADA supplied to record the profit upon withdrawal
4. Remove Liquidity
  - Pulls OADA+ADA tokens out of the pool, and, in proportion to the withdrawn LP it holds, takes account of the total sum received (i.e. if it holds 100 LP and supplied 1000 ADA + OADA, then the withdrawal of 50 LP that gets the system 600 ADA+OADA implies the system has made a 100 ADA profit)
  - Immediately burns the OADA pulled out of the pool.

Note: Given mathematical perfection is not possible with floating point arithmetic and a negligible but non-zero splash protocol fee – some error tolerance must be had, and the system is set to have a *slightly tighter* range for deposits. Therefore there is a range of current price to target price that is impossible to rebalance to without the system taking on a loss for doing so. It is incredibly precise and unreasonable to expect to be able to be induced on the open market for long, but it can cause the system to be unable to perform the add liquidity action temporarily. This is expected behavior, and the OADA controller simply waits until the price moves to a price from which it can rebalance the price higher without taking on a loss. (i.e. system has to profit enough by repegging it from slightly lower to push it slightly into territory where it loses money by pushing the open market price into – as it cannot make a losing trade)

6.7.1. OADA Splash DEX Rule

Introduced in	1.0
Last Revision	1.0
Package	Base
Source	Not Open Sourced
Audits	None Currently
Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH
As the DEX AMO is most complex, this is in turn, as one could expect, the most complex rule, one that enables each of the 4 actions that must be performed.	

6.8. Donation Strategy

Introduced in	1.0
Last Revision	1.0

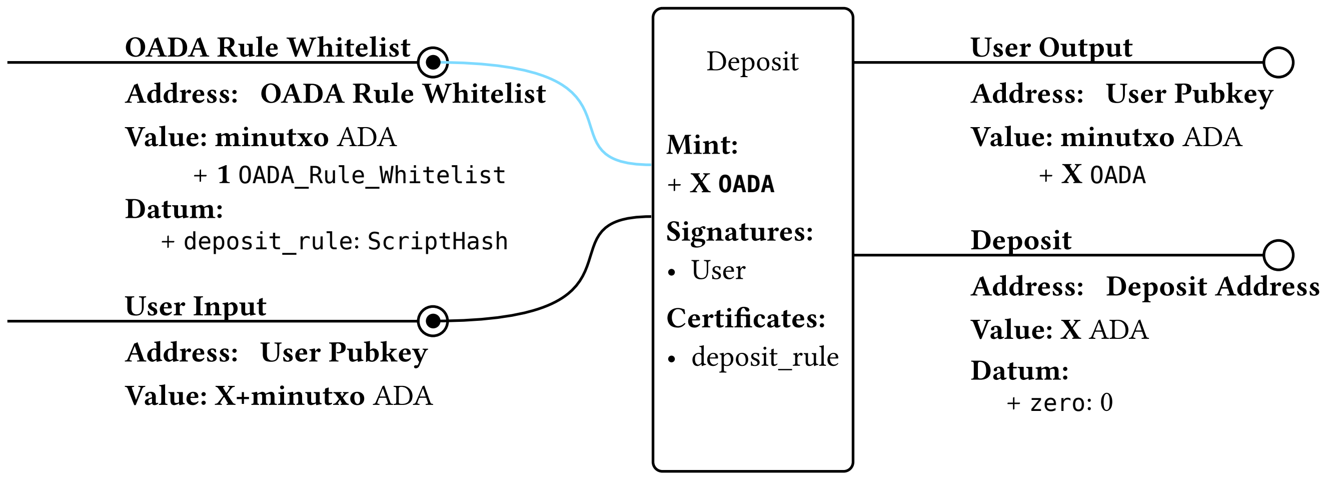
Package	Core
Source	https://github.com/OptimFinance/clean-code
Audits	https://github.com/OptimFinance/clean-code
Plutus Version	2
Unparameterized Script Hash	HASH
Deployed Script Hash	HASH
The only Strategy in the system, its purpose is to allow anyone to donate ADA to the system to count it as profit. Mostly used for working around Catalyst, but also given as the simplest example of a strategy. A note on the terminology used, as it doesn't have an associated Rule, it is considered a Strategy. If it were to be enriched with a Rule, to burn donated OADA (of which donations are currently unaccepted) then it would be considered an AMO.	

7. System Transactions

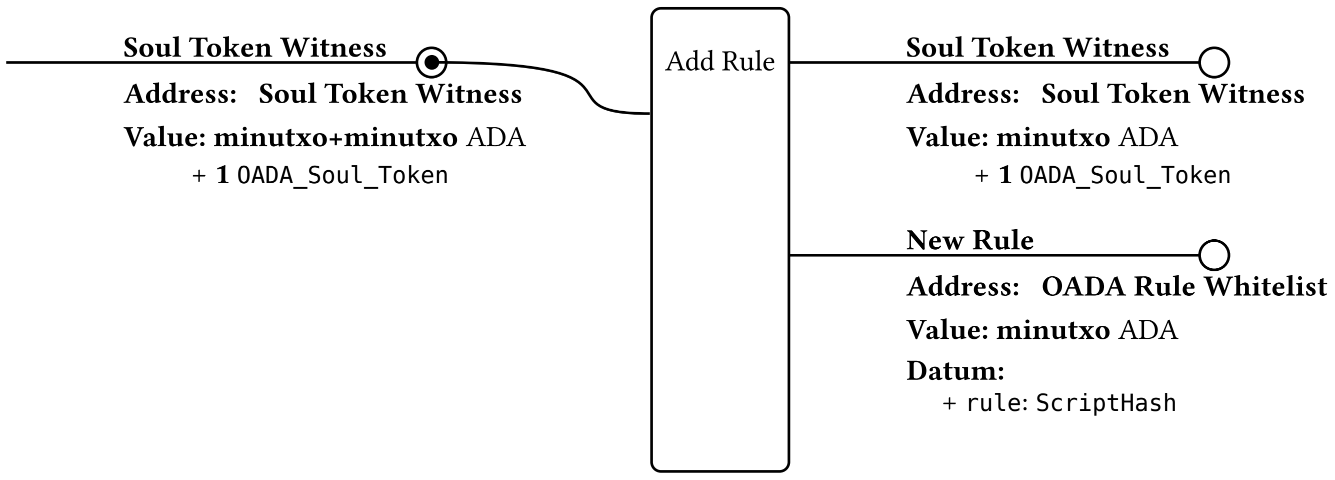
In this section we present the totality of the transaction schemas used in the system's operation. We present the *minimal* version of each transaction, as we attempted to make it possible to include other contracts and transfers while performing the system operations – nevertheless do verify in code before assuming that one's use case is permitted to be composable with a given system.

Some liberties are taken in order to present the system on a conceptually coherent level, and simplifications made. This is a catalogue of all of the actions performed by the system and not a full specification of them.

7.1. Mint OADA

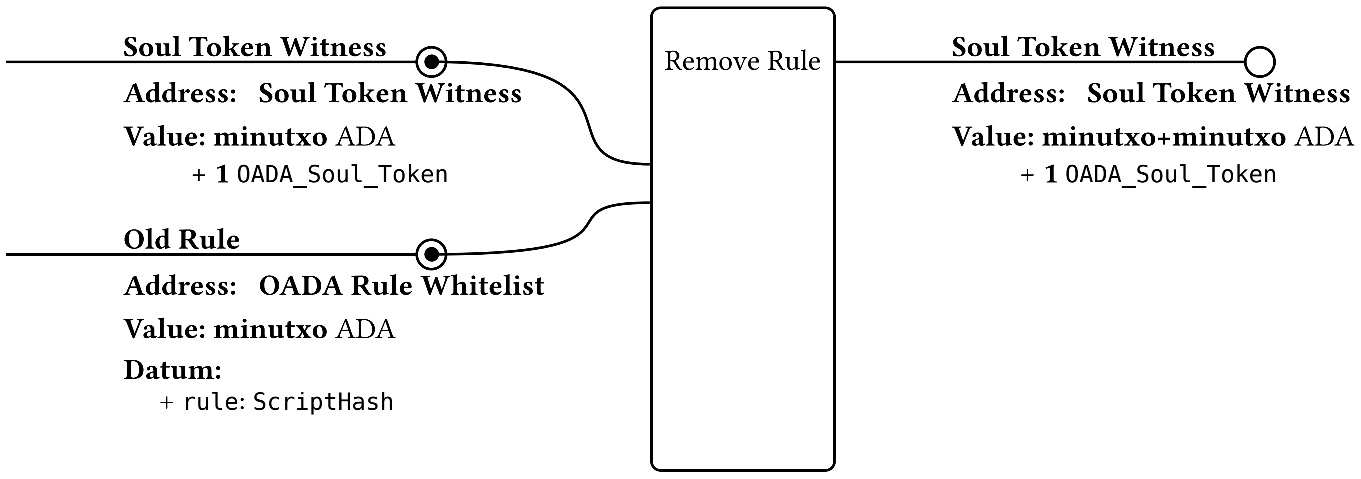


7.2. Add OADA Rule



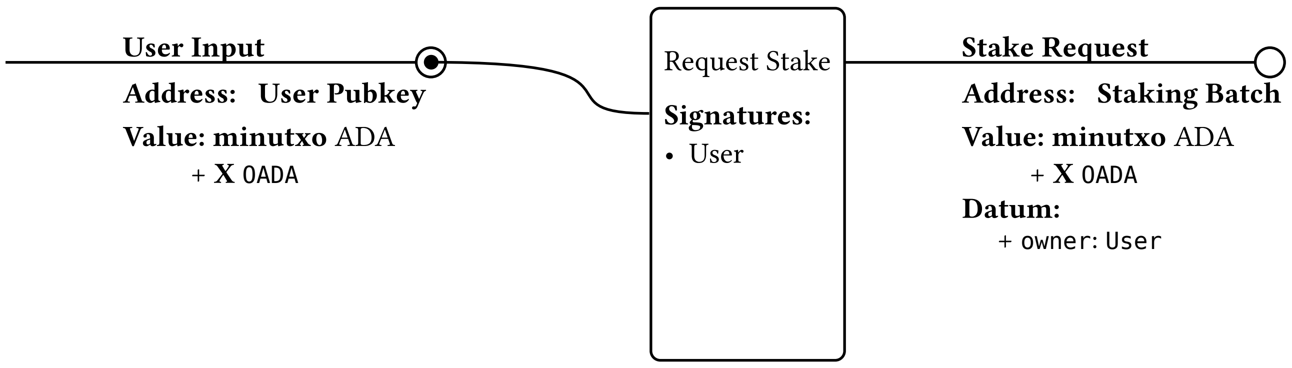
Note: Sctipt is inlined

7.3. Remove OADA Rule

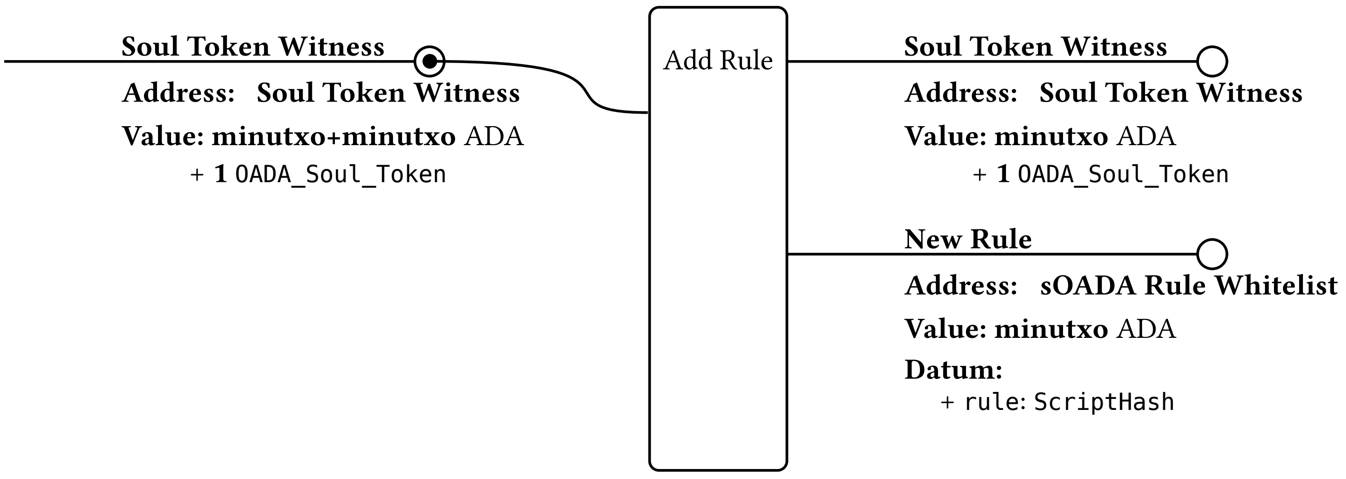


Note: Sctipt is inlined

7.4. Request Stake of OADA

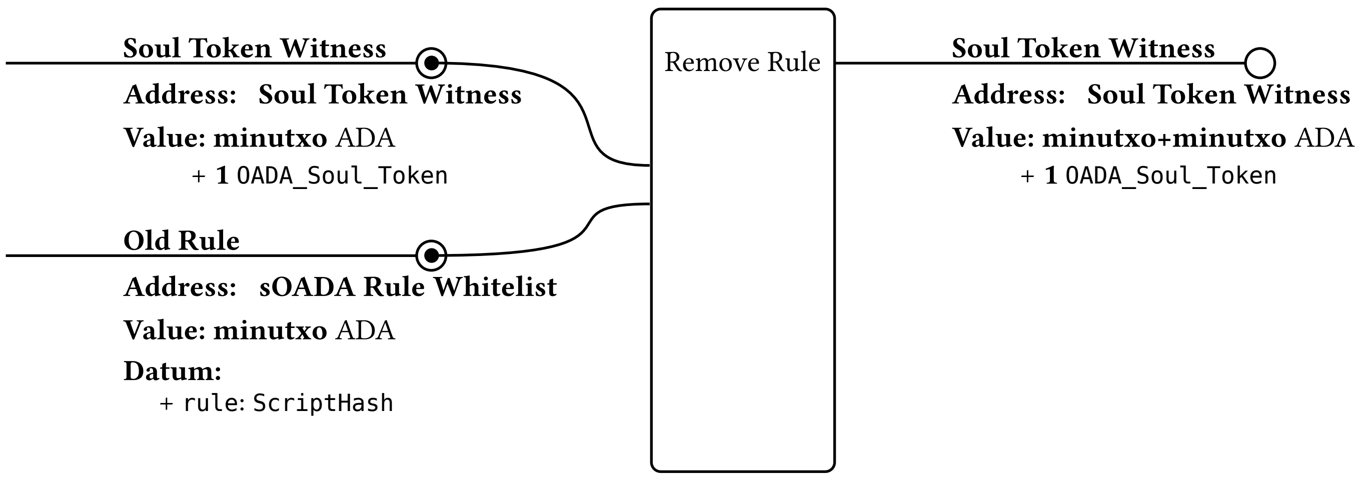


7.5. Add sOADA Rule



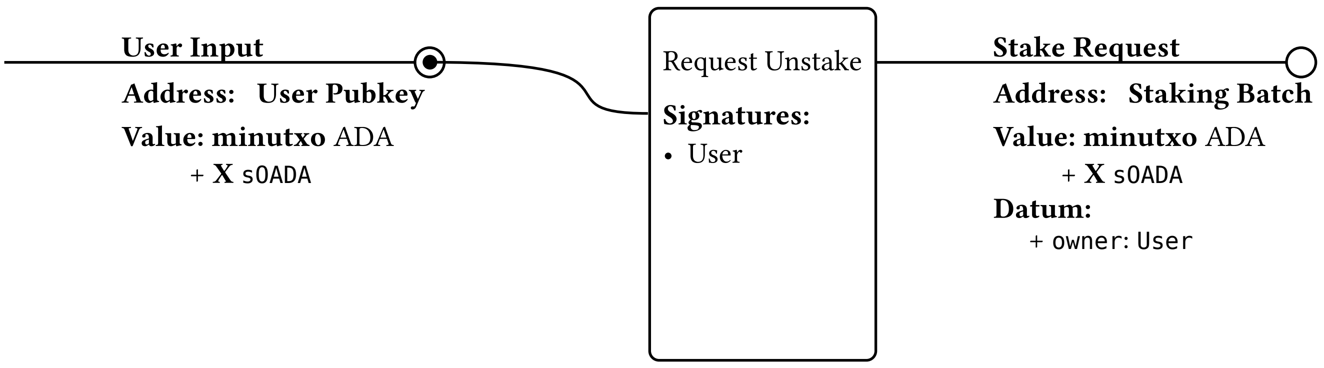
Note: Sctipt is inlined

7.6. Remove sOADA Rule

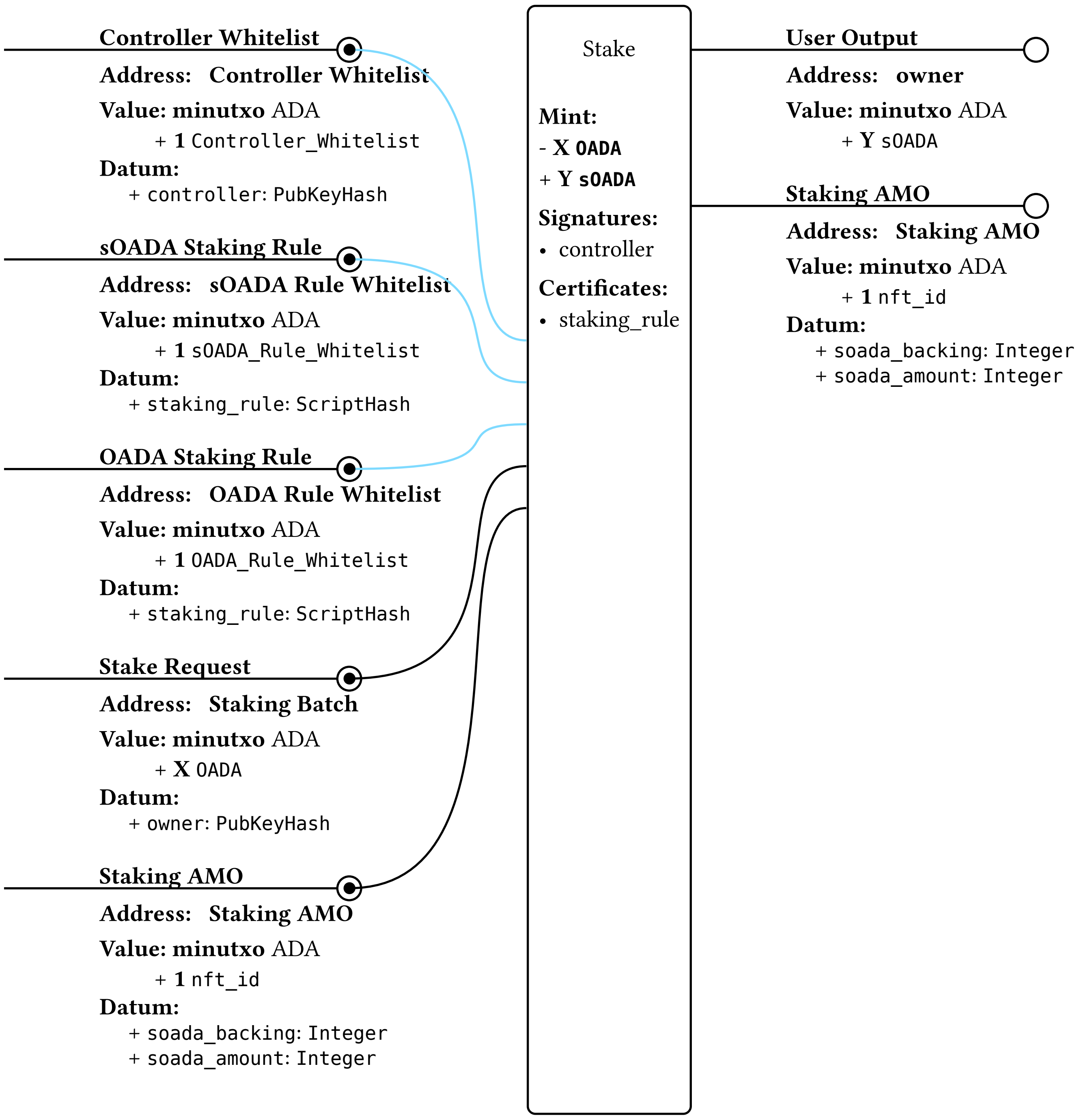


Note: Sctipt is inlined

7.7. Request Unstake of sOADA

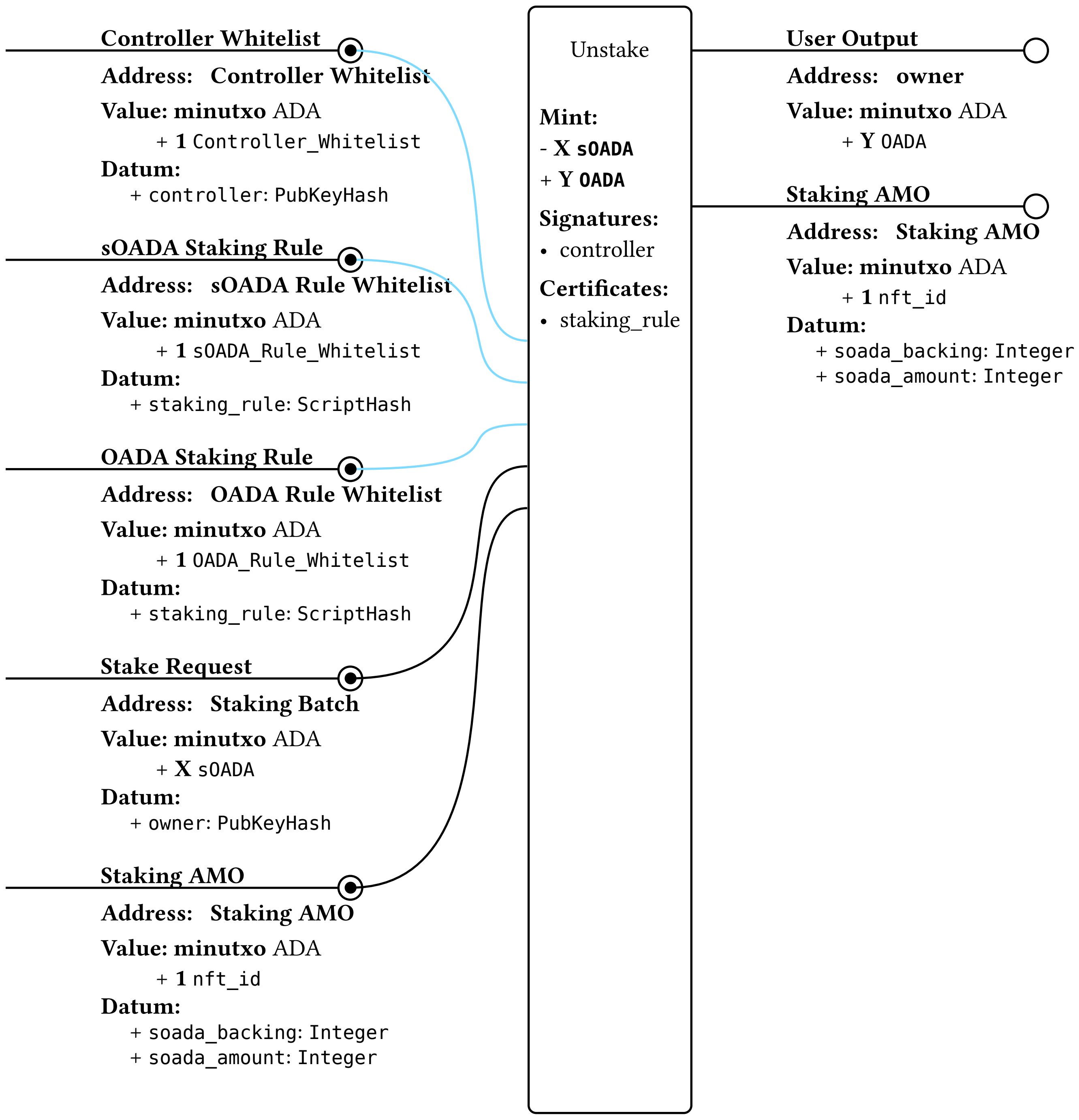


7.8. Stake OADA for sOADA



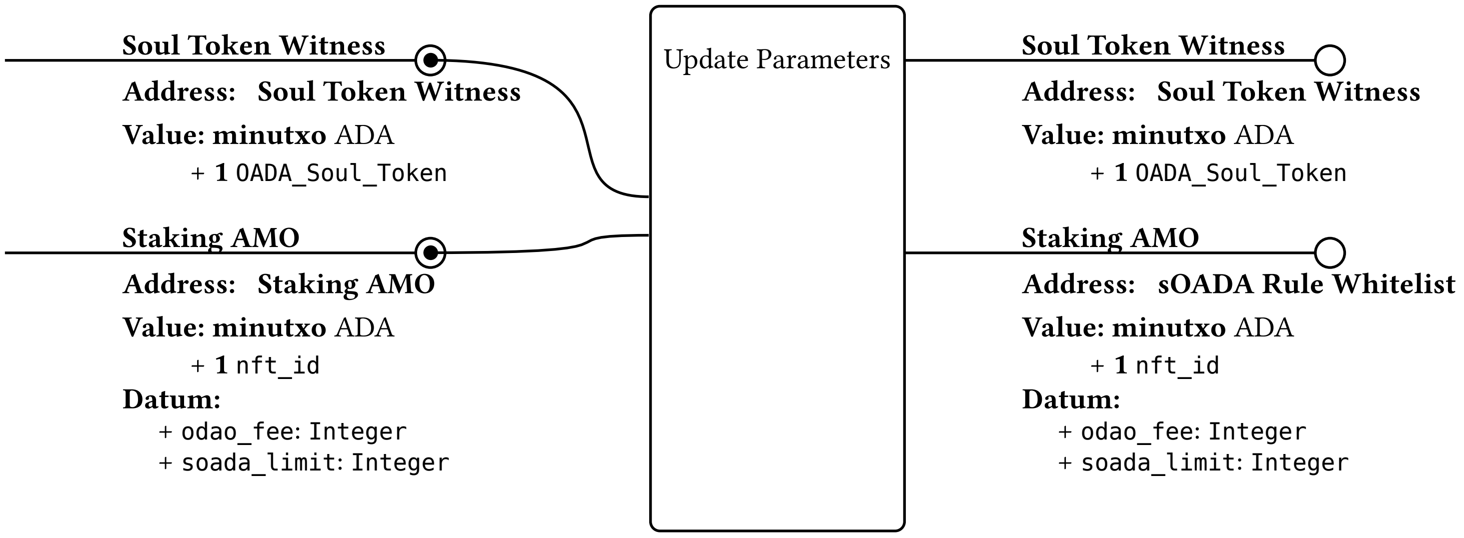
Note:
Y = X*sOADA_backing/sOADA_amount
Updates the ‘soada_backing’ and ‘soada_amount’ to reflect the action
The ‘staking_rule’ is the same script for OADA and sOADA

7.9. Unstake sOADA for OADA

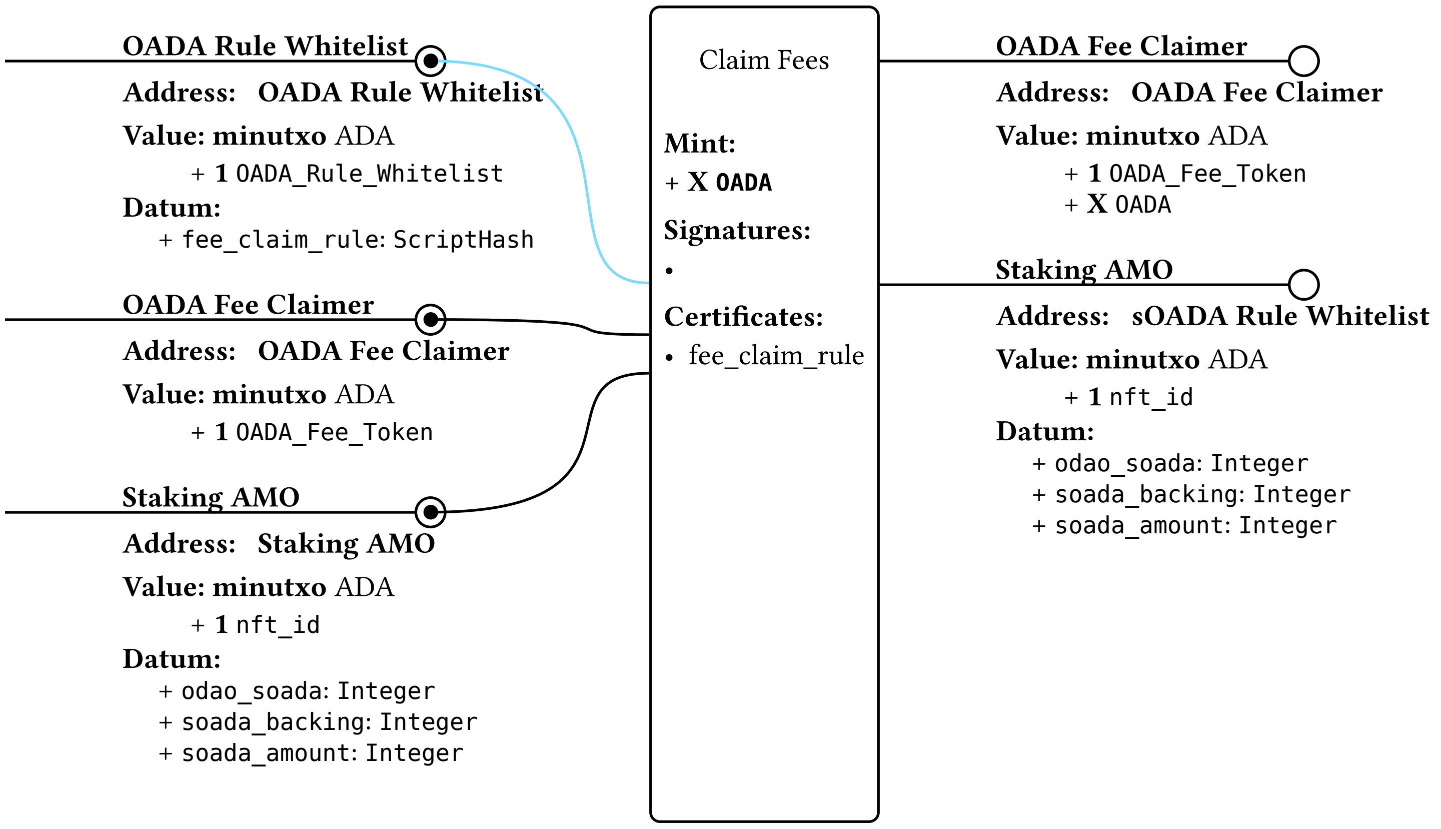


Note:
Y = X*0.999*sOADA_amount/sOADA_backing
Updates the ‘soada_backing’ and ‘soada_amount’ to reflect the action
The ‘staking_rule’ is the same script for OADA and sOADA

7.10. Perform Staking AMO Paramerter Update

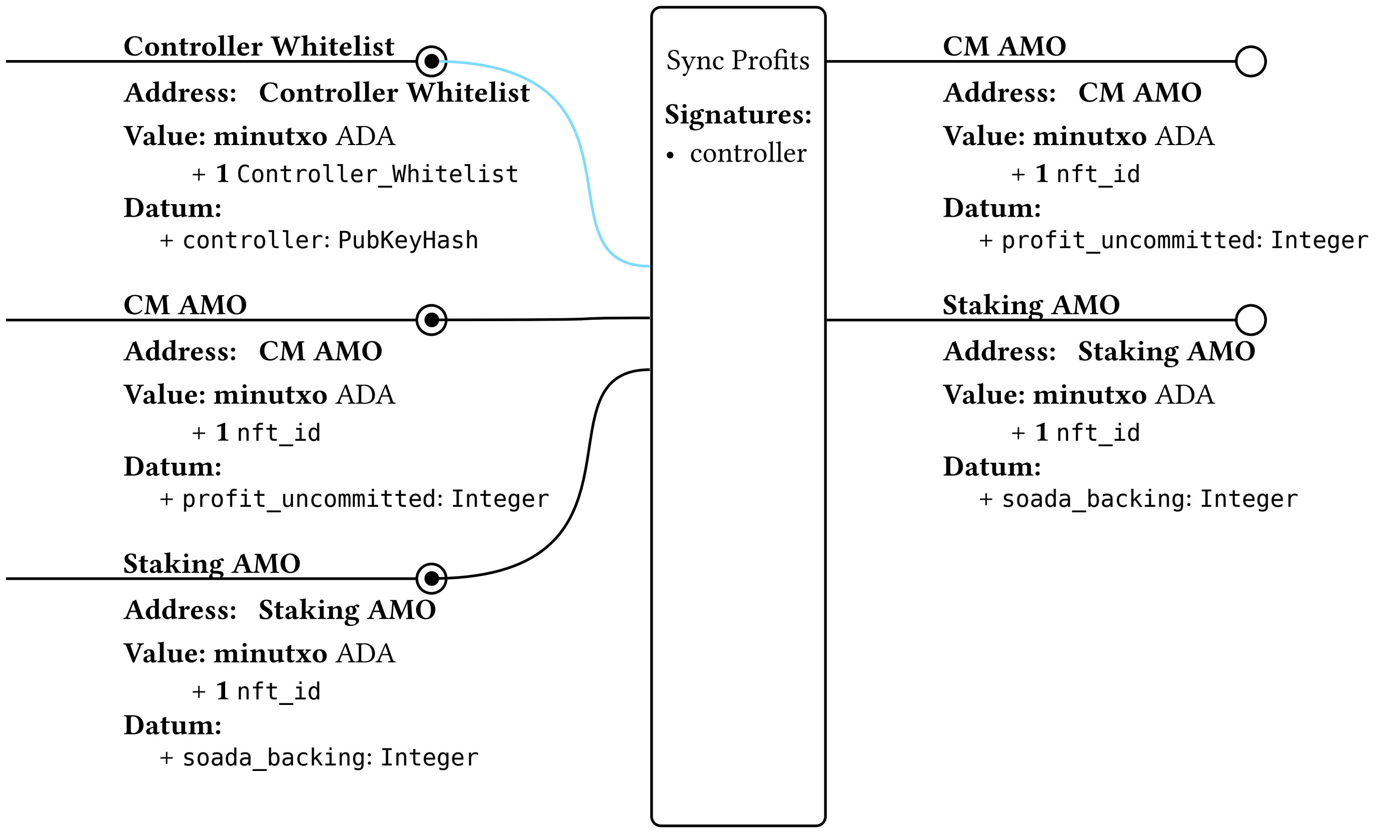


7.11. Claim ODAO Fees



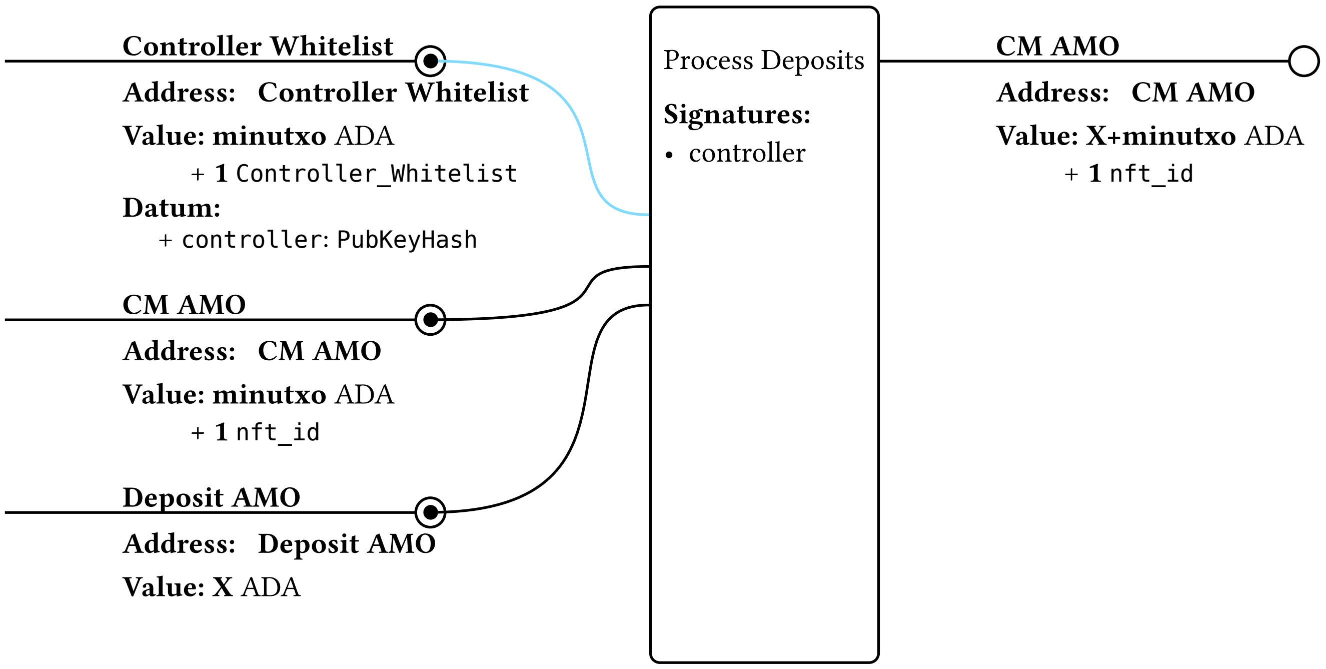
Note:
X = (input ‘odao_soada’ - output ‘odao_soada’) * soada_backing / soada_amount
Importantly, upon fee claim, the odao skips the 0.1% unstake fee present on regular staking

7.12. Sync Profits with Staking AMO



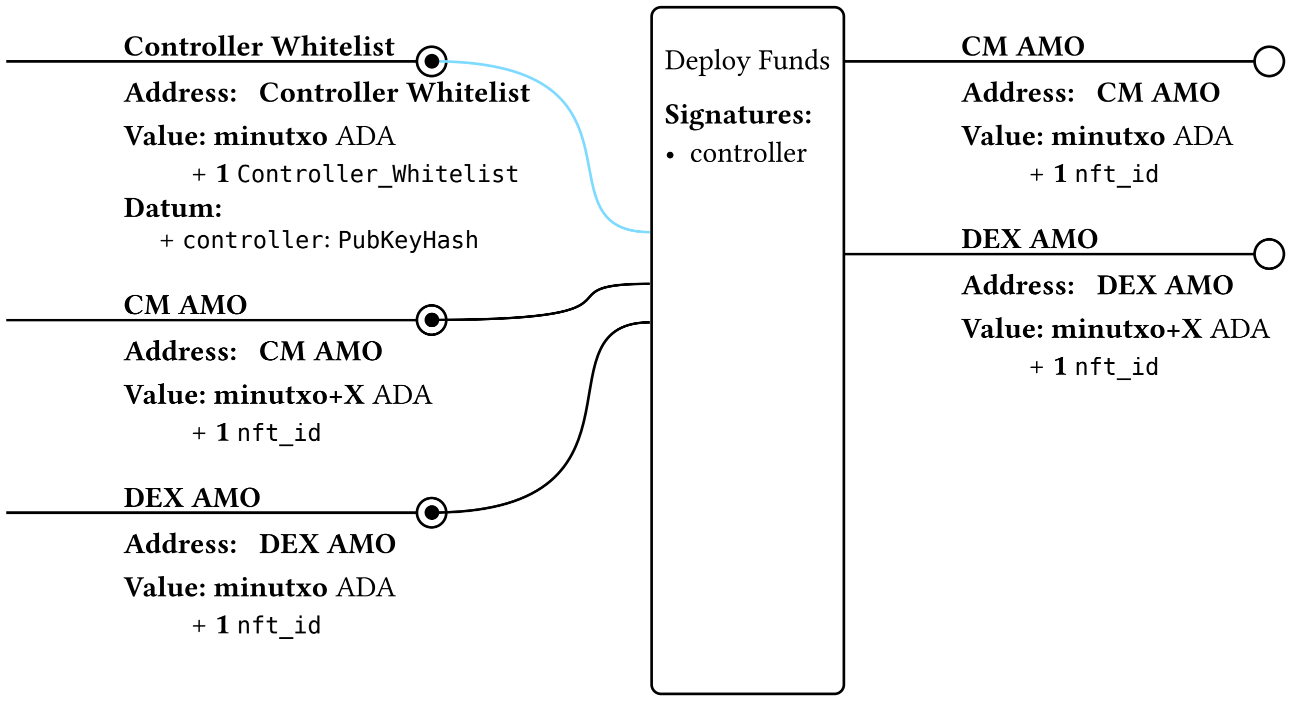
Note: profit_uncommited is added to soada_backing

7.13. Process Deposits in Collateral Management AMO

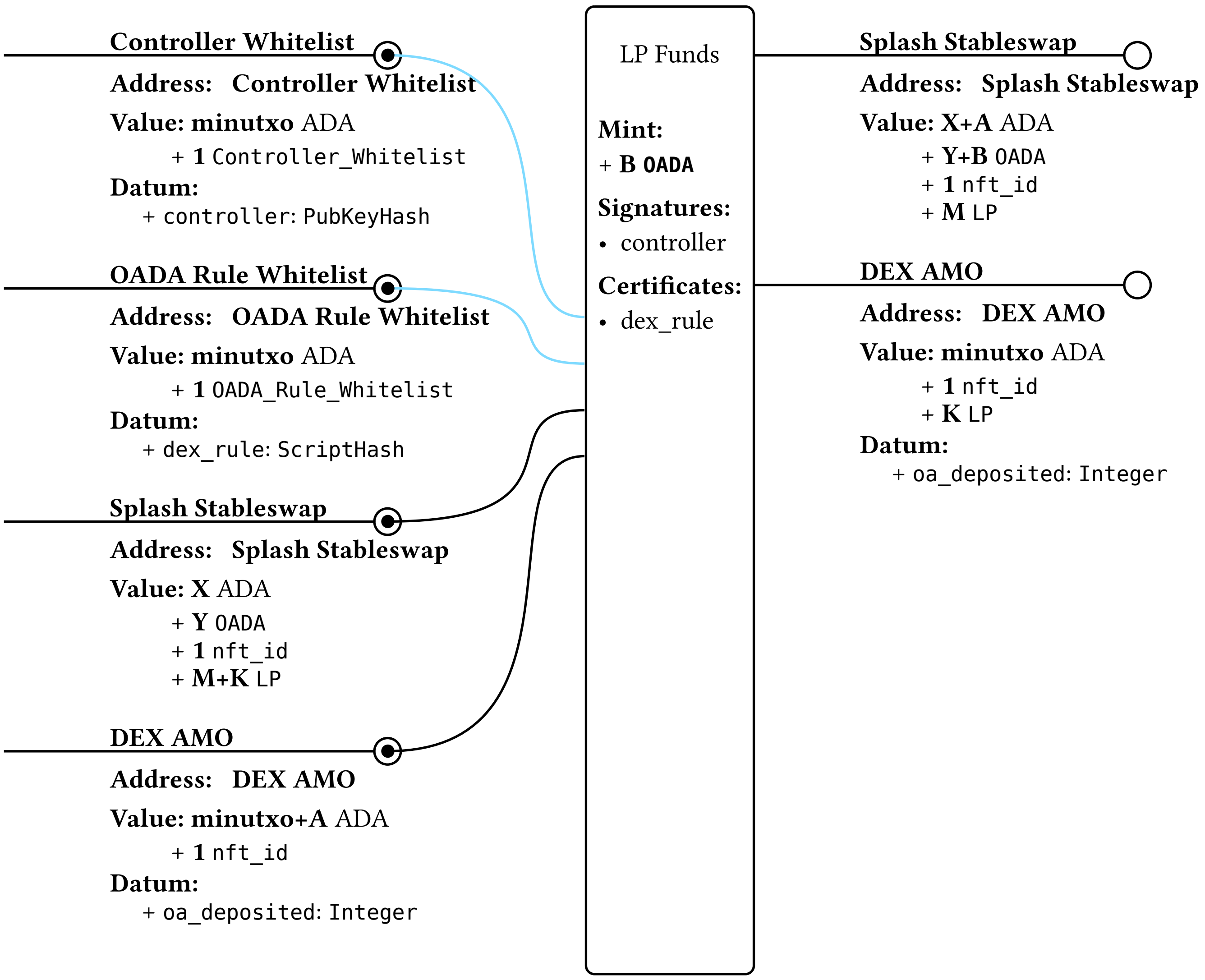


Note: Can process multiple dozen deposits at a time

7.14. Deploy Funds into the Splash DEX AMO

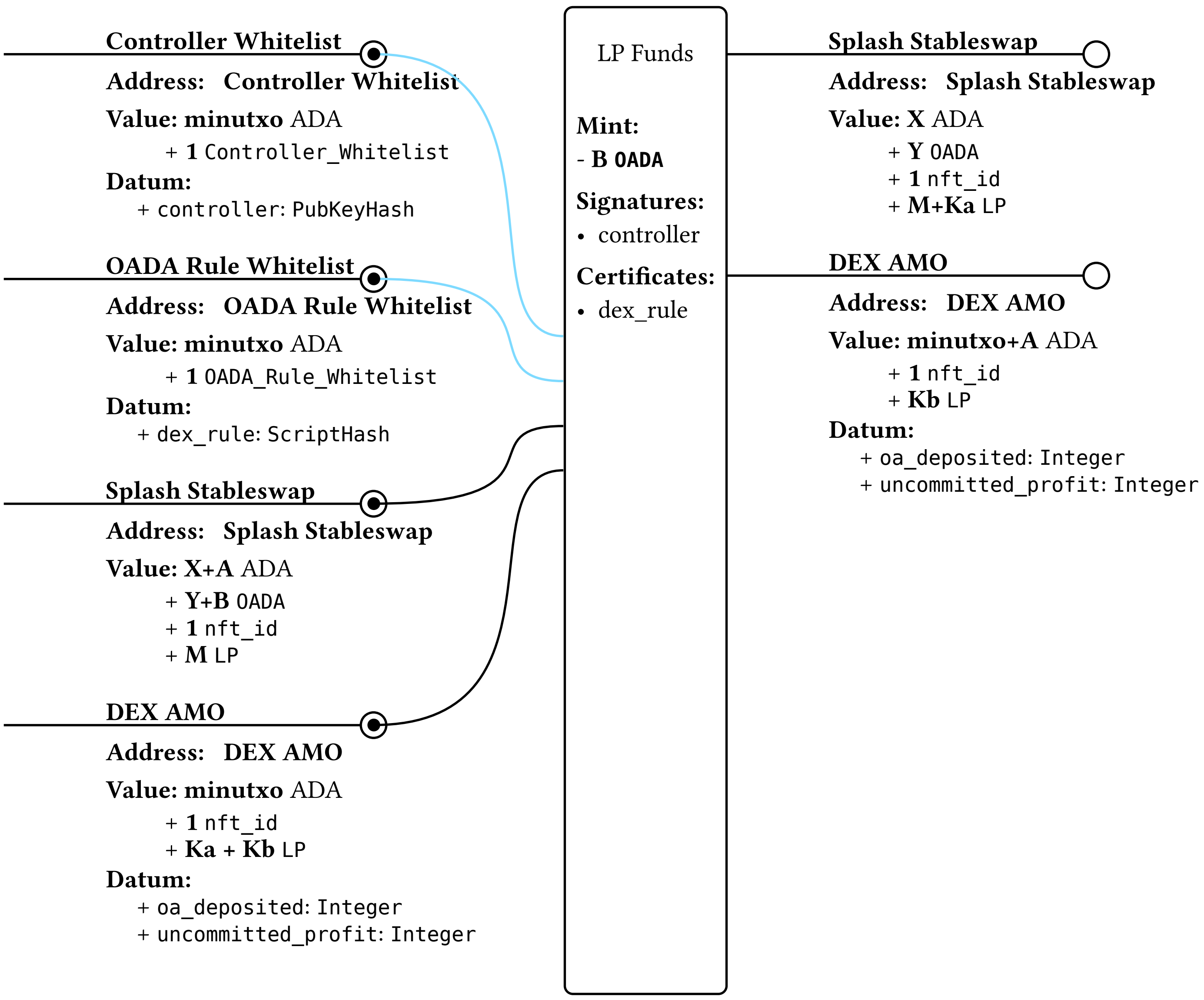


7.15. Deposit ADA with OADA into Splash Stableswap



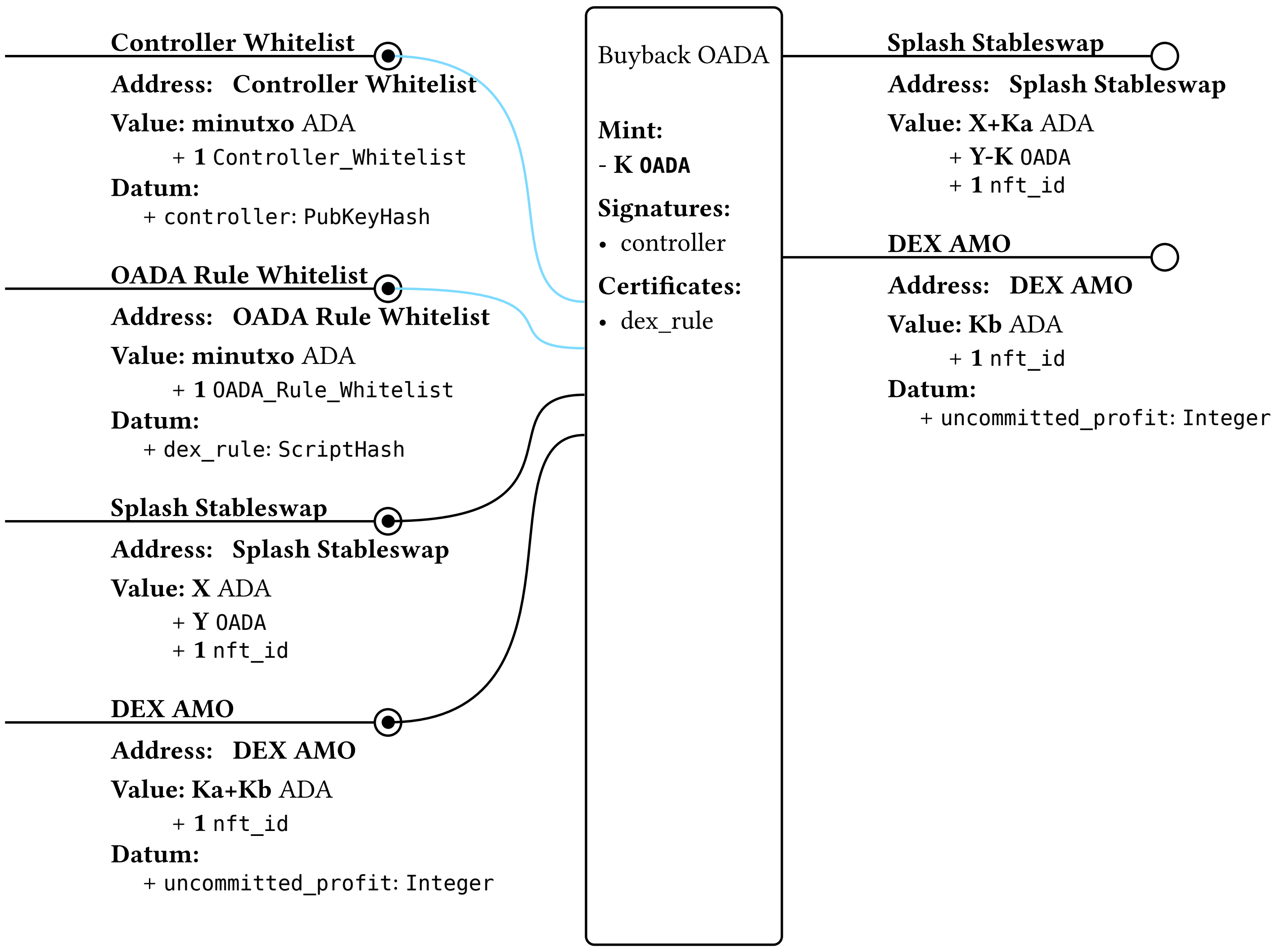
Note:
deposit_guard <= X / Y
deposit_guard >= Y / X
deposit guard ensures the deposit happens between 0.999 and 1.001
A+B is added to oa_deposited

7.16. Withdraw LP from the Stableswap



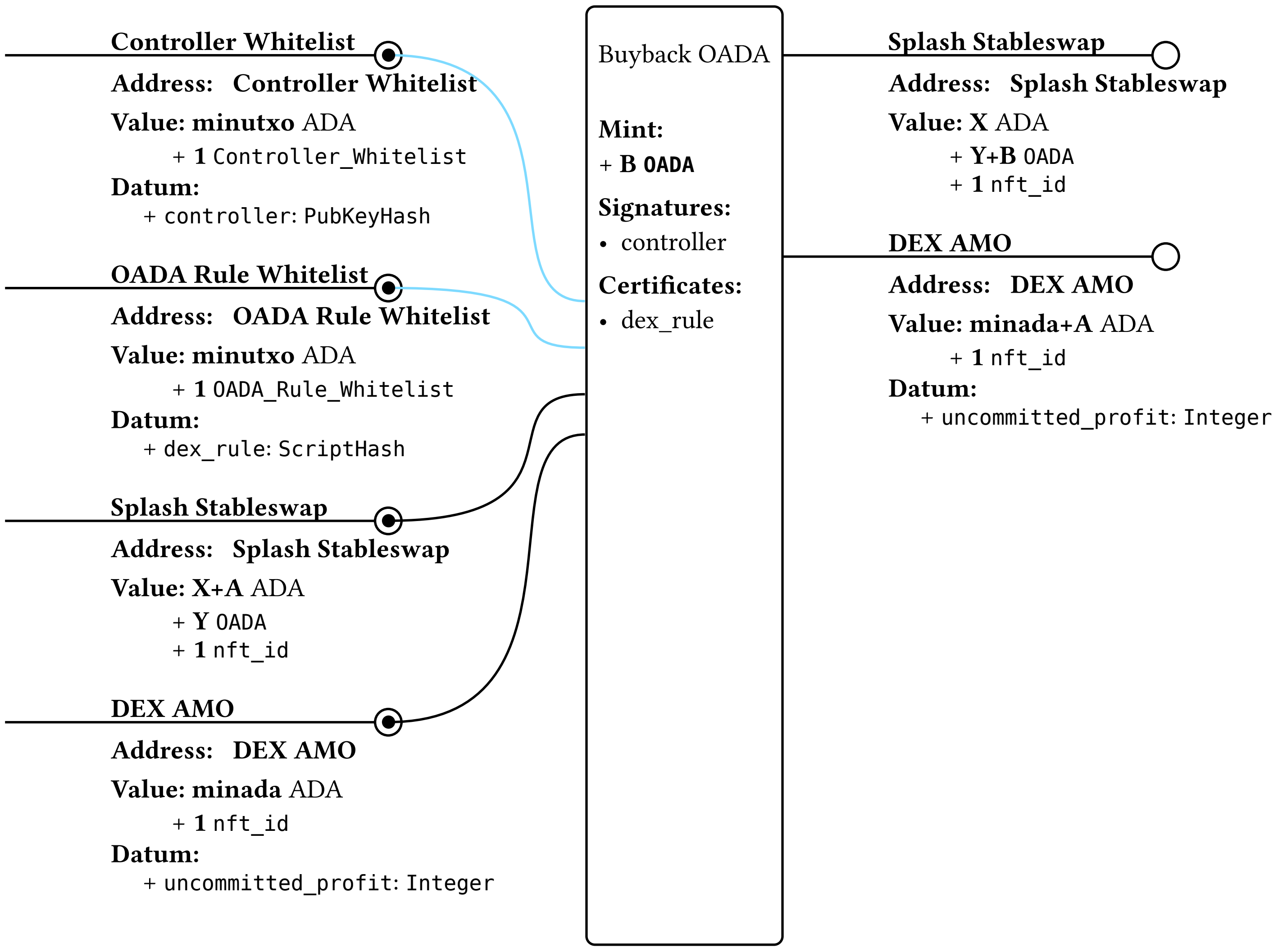
Note:
A+B is subtracted from oa_deposited
profit += (withdrawn - deposited) * (Ka / (Ka + Kb))

7.17. Protect Downside Peg on the open market



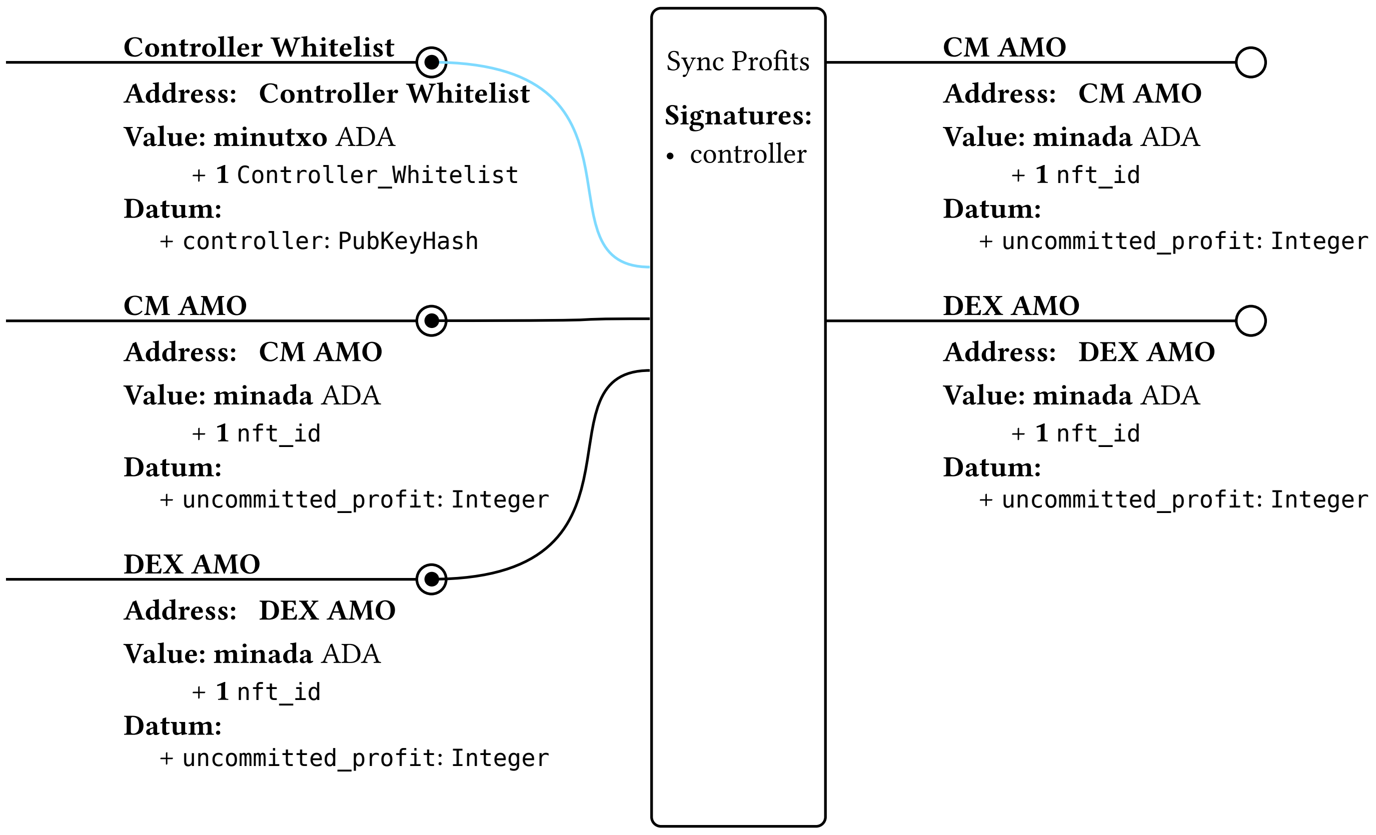
Note:
K >= Ka
profit += K - Ka

7.18. Protect Upside Peg on the open market

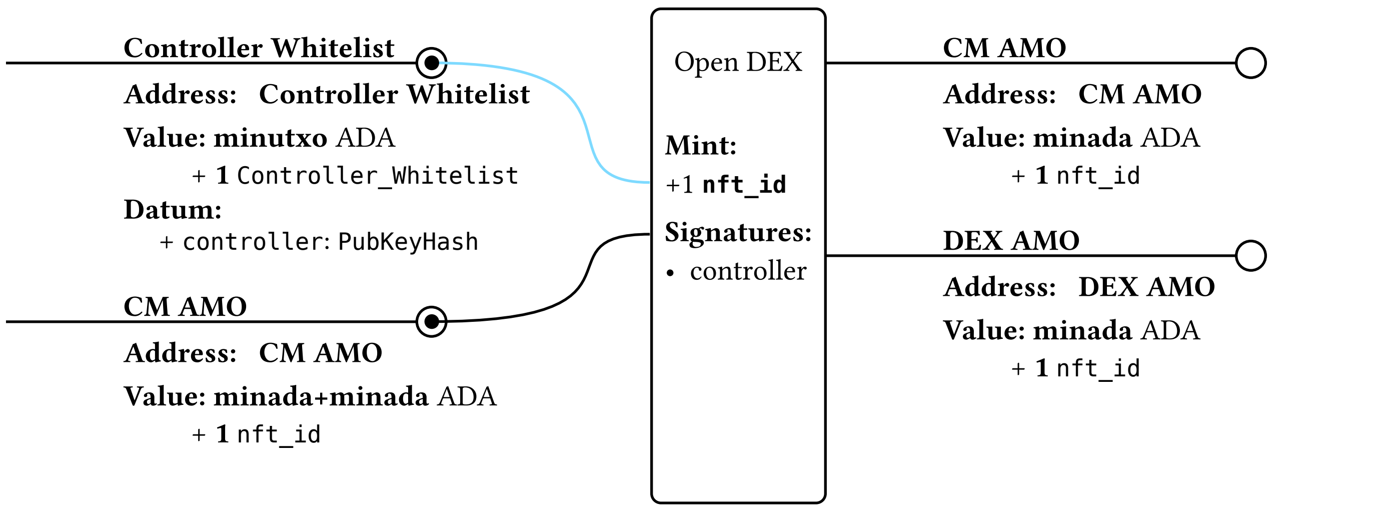


Note:
A >= B
profit += A - B

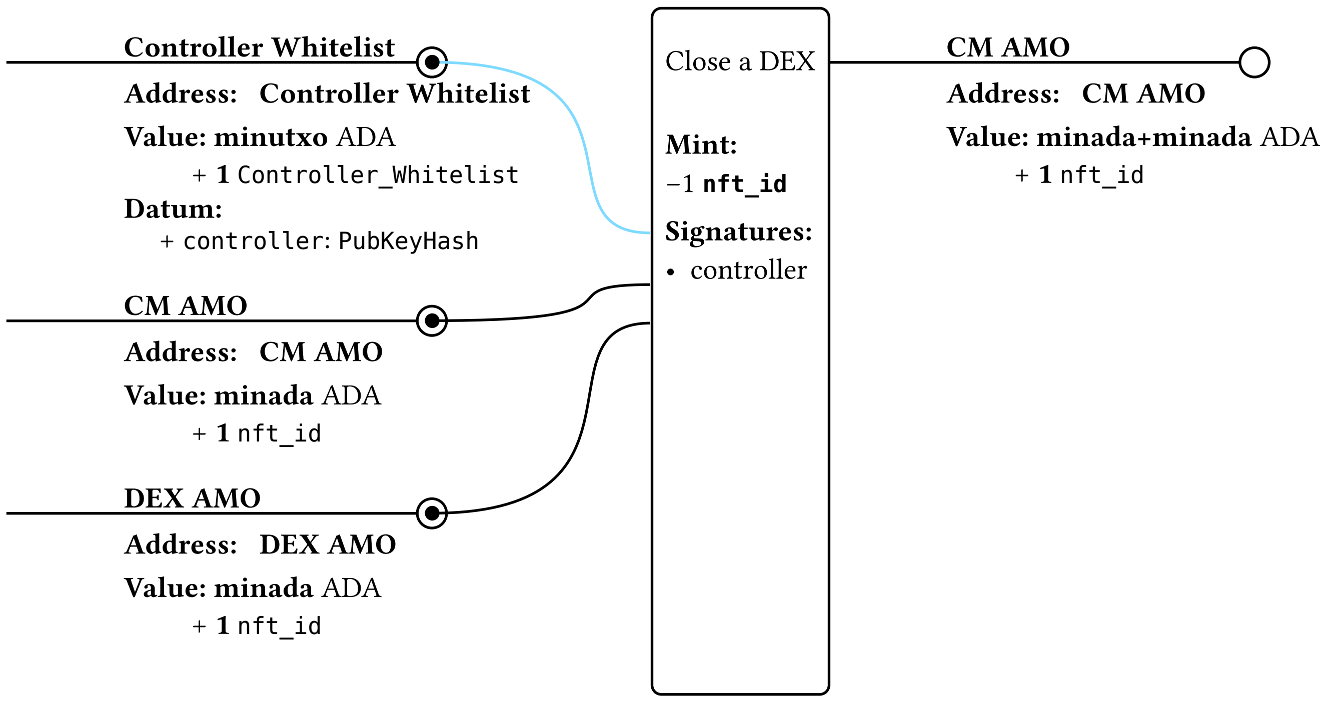
7.19. Sync DEX AMO Profits with the CM AMO



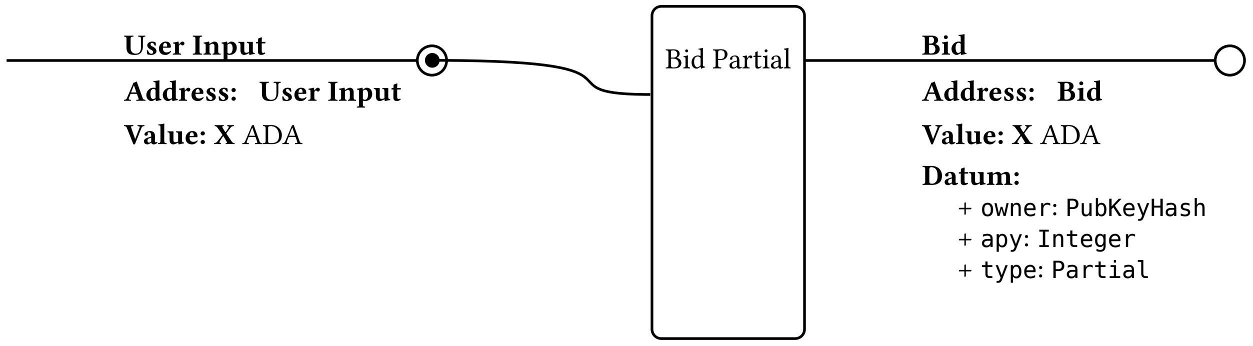
7.20. Open a DEX AMO



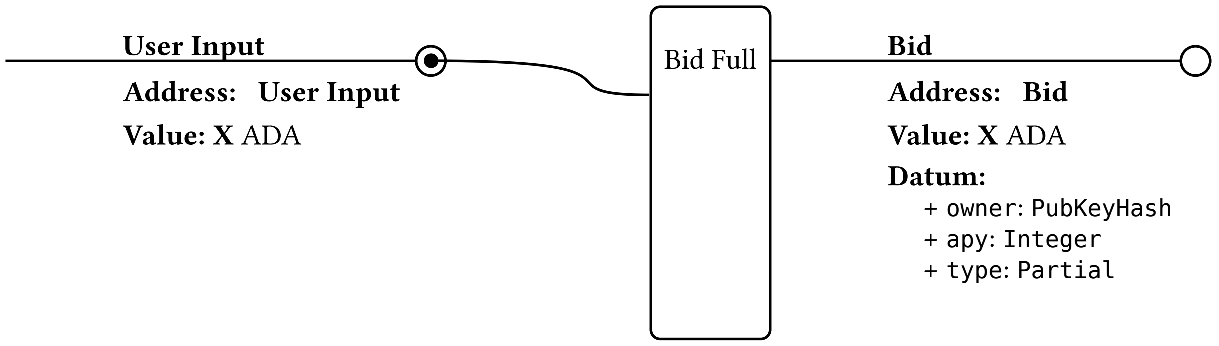
7.21. Close a DEX AMO



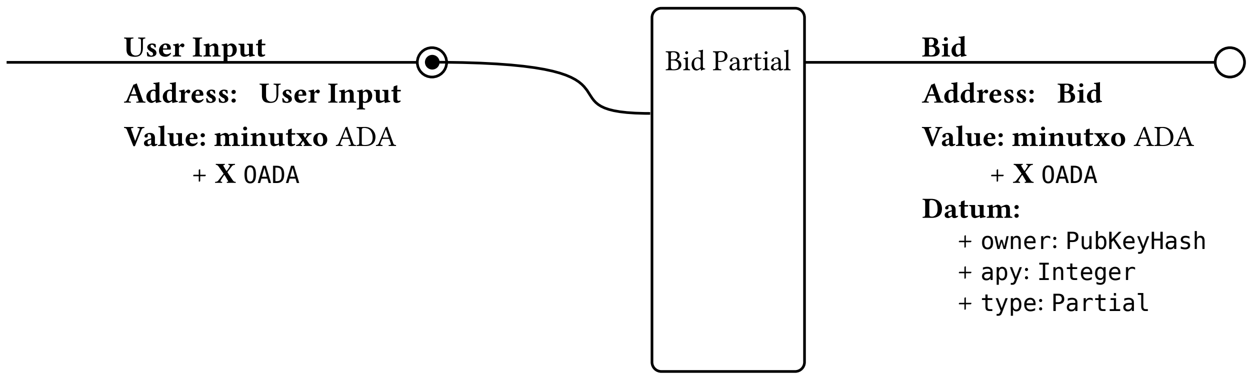
7.22. Partial Fill Bid ADA in the Staking Auction



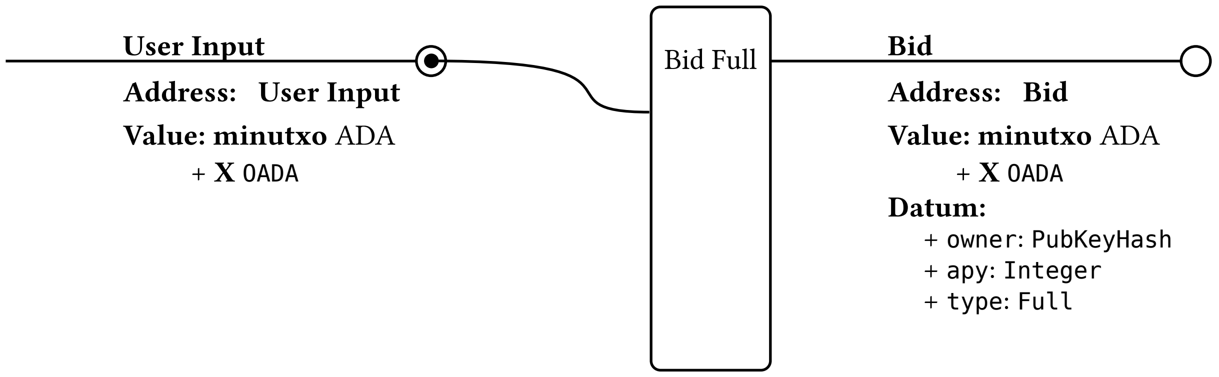
7.23. Fill or Kill Bid ADA in the Staking Auction



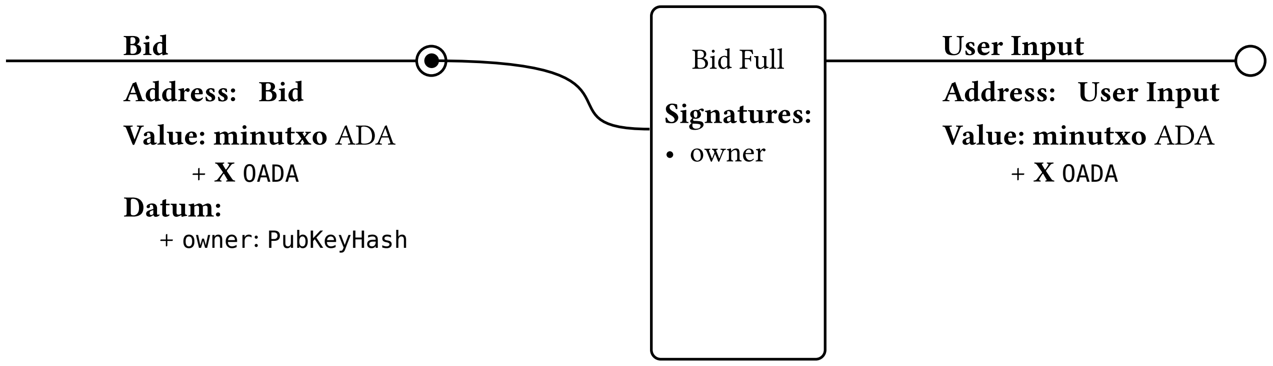
7.24. Partial Fill Bid OADA in the Staking Auction



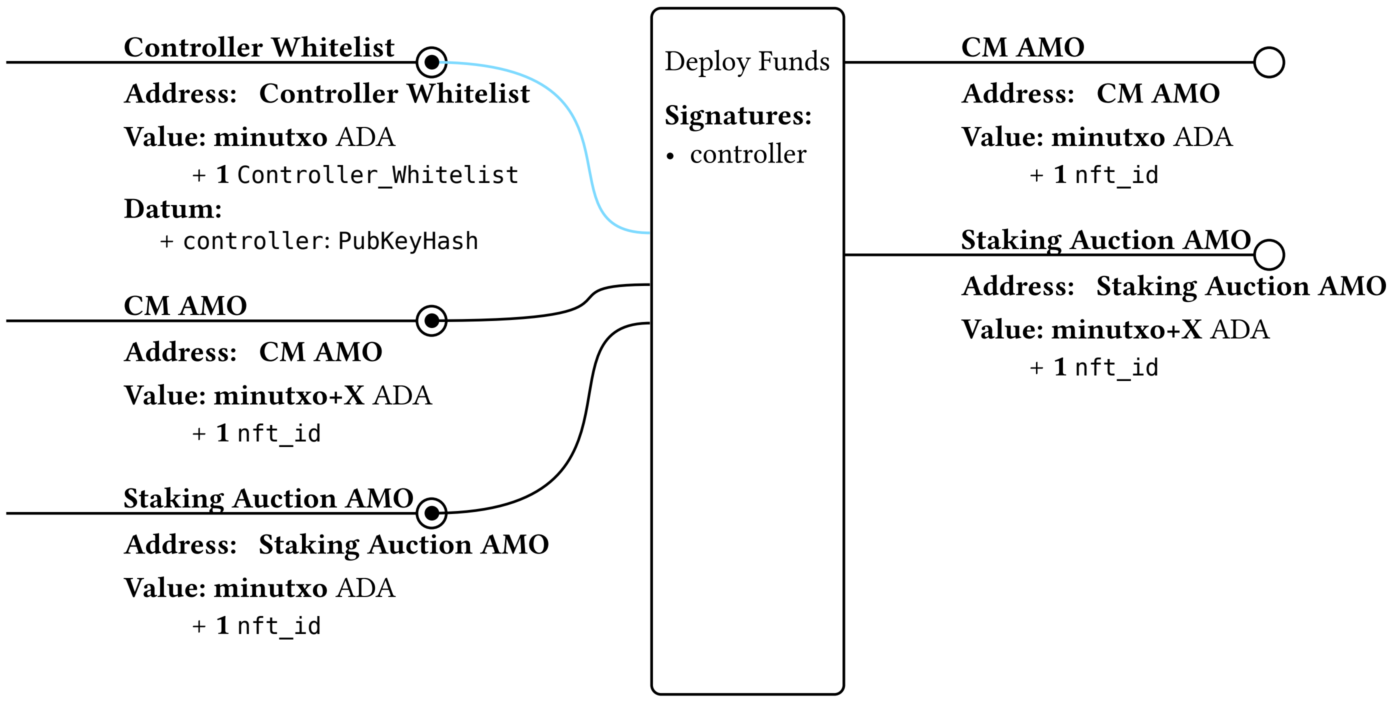
7.25. Fill or Kill Bid OADA in the Staking Auction



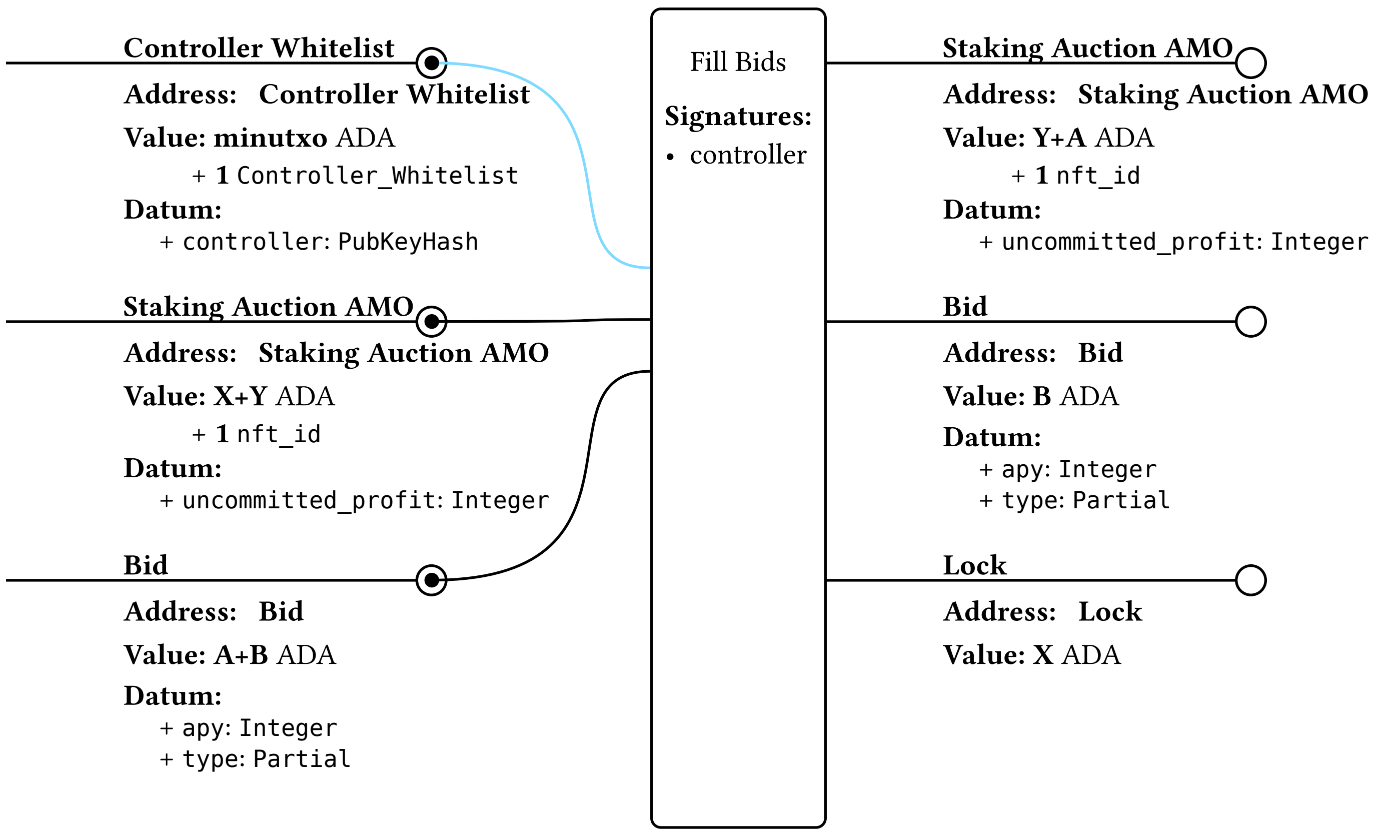
7.26. Cancel a bid



7.27. Deploy Funds into the Staking Auction AMO

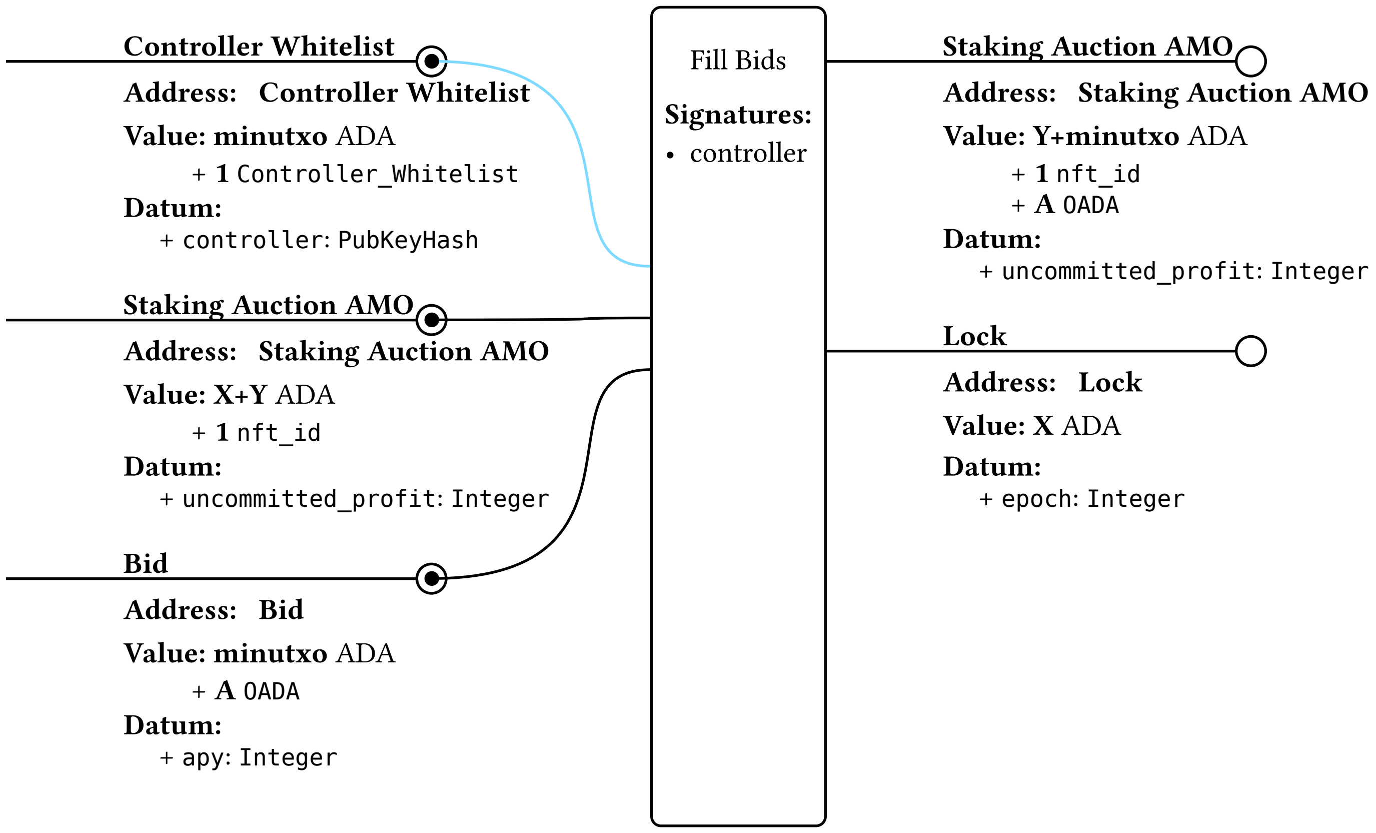


7.28. Partial Fill a Partial Bid



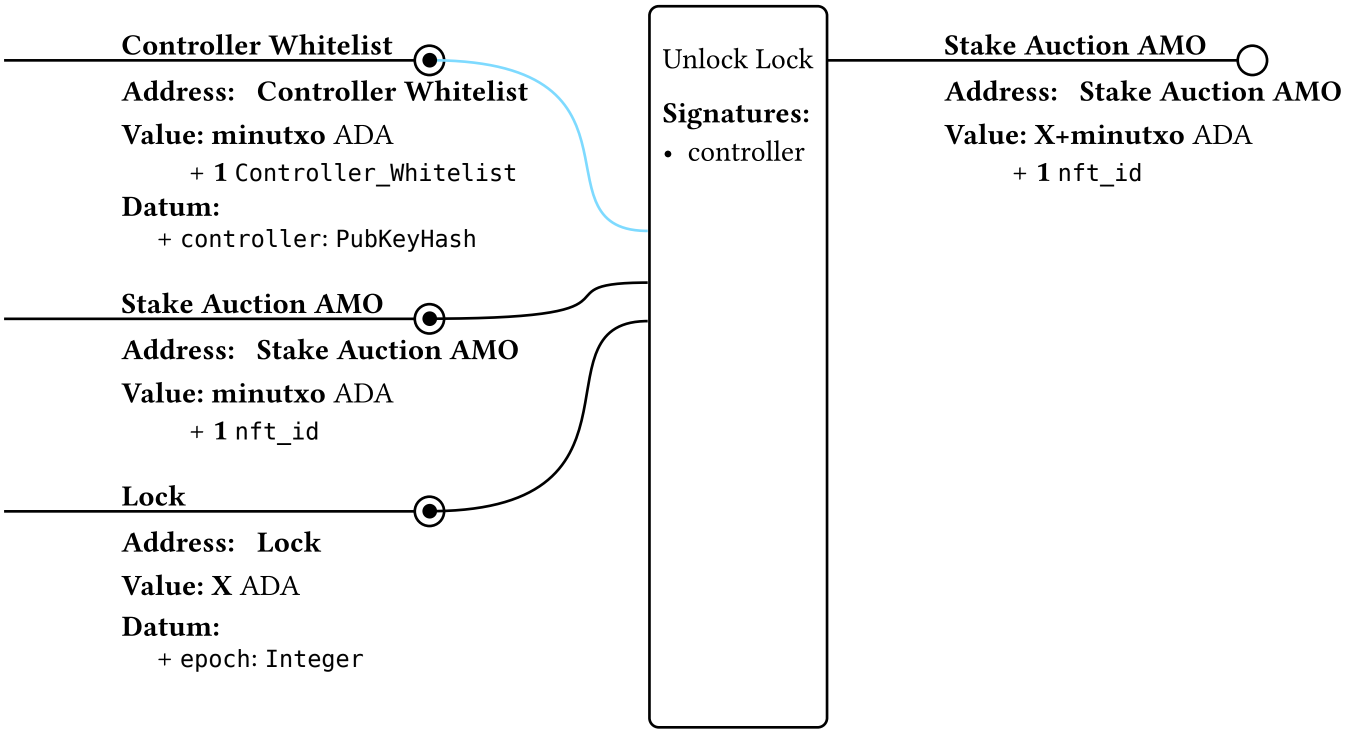
Note:
Can partial/fill multiple bids at once
X = B / (apy/73)
Lock has the current epoch in datum
uncommitted_profit += A

7.29. Fill a Fill or Kill Bid



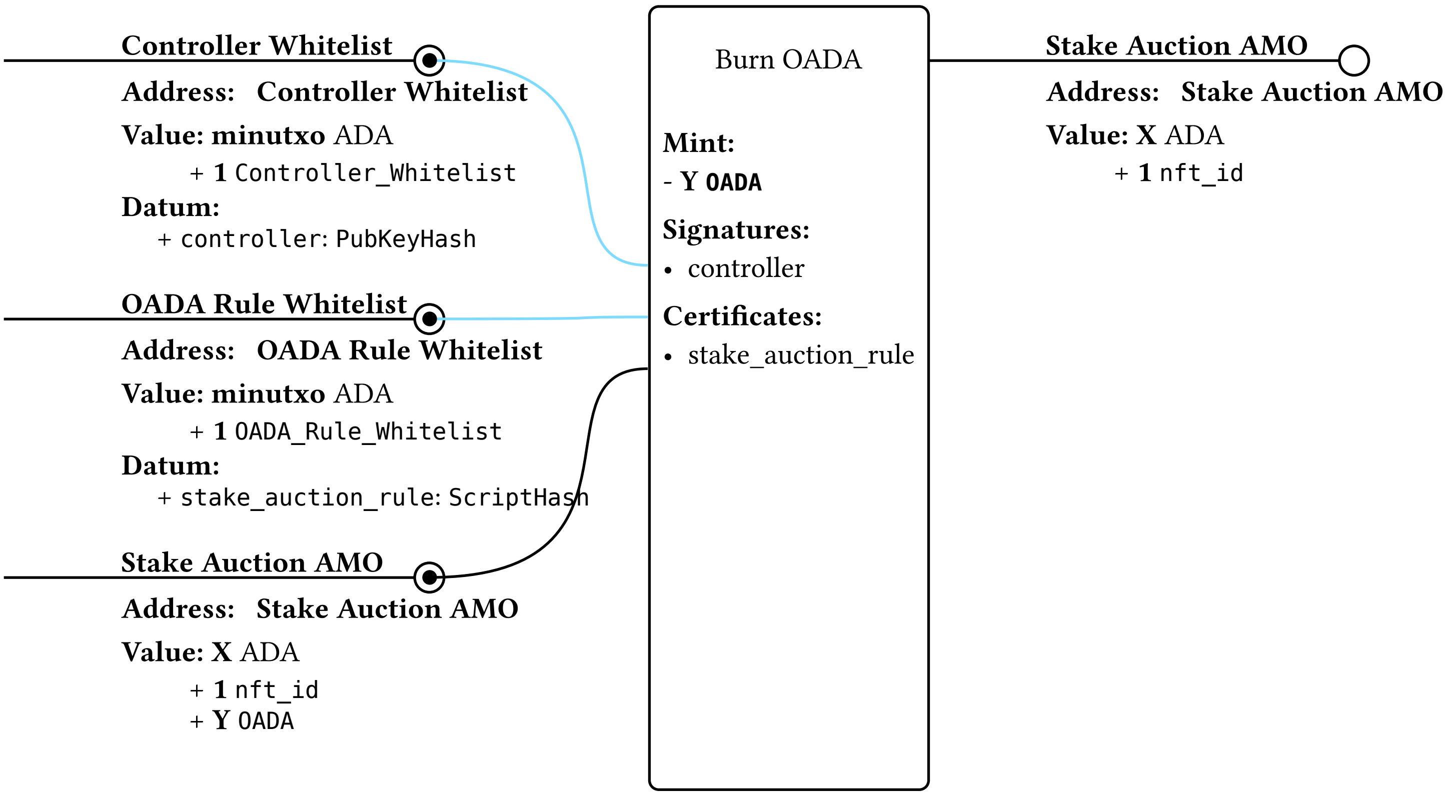
Note:
Can partial/fill multiple bids at once
X = B / (apy/73)
Lock has the current epoch in datum
uncommitted_profit += A

7.30. Process expired Stake Locks



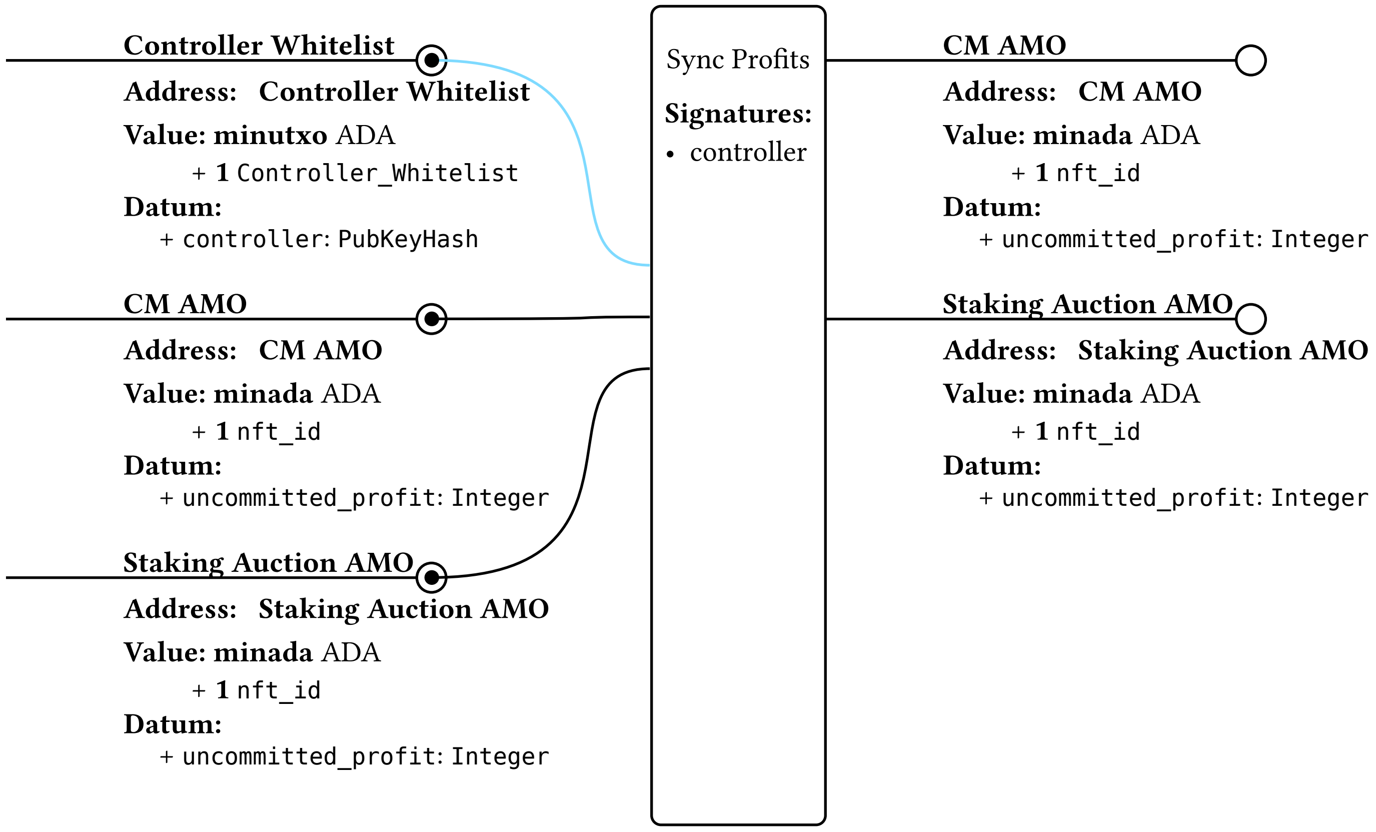
Note: Can unlock multiple at a time

7.31. Burn OADA from Bids

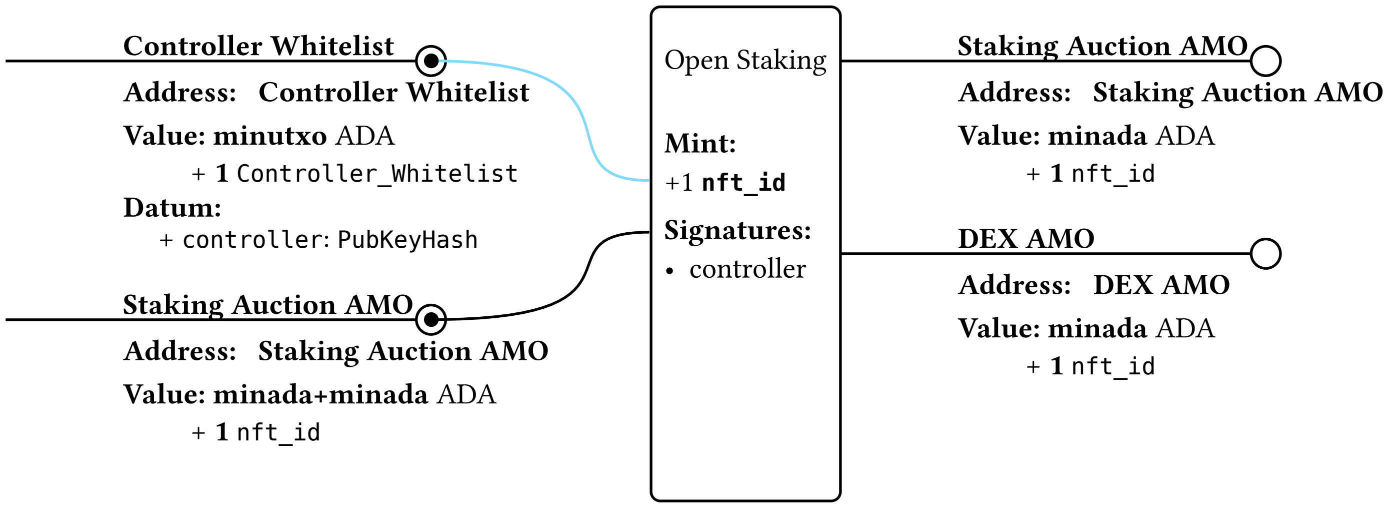


Note: Can unlock multiple at a time

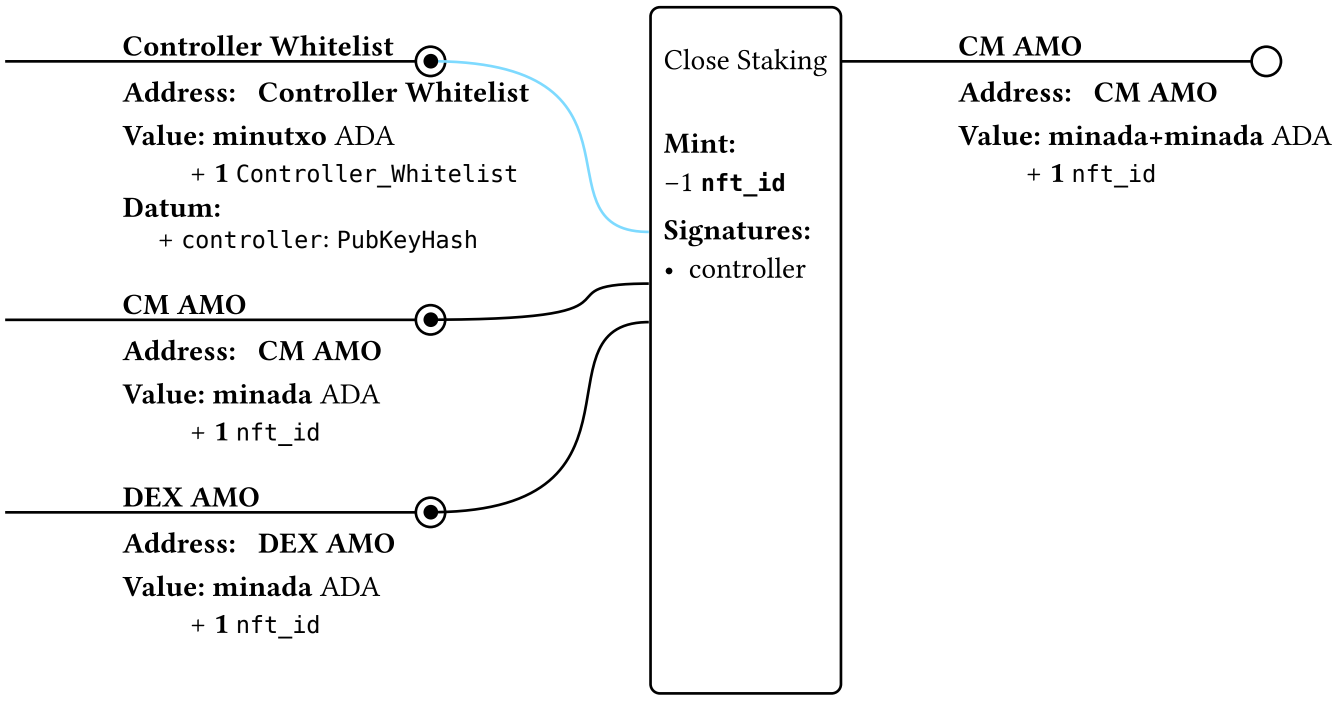
7.32. Sync Staking AMO Profits with the CM AMO



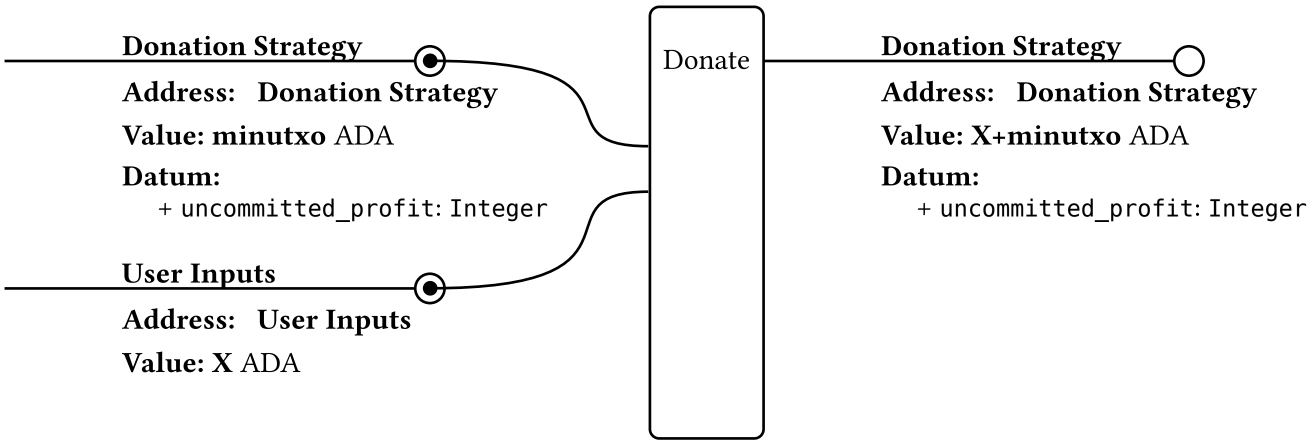
7.33. Open a Staking Auction AMO



7.34. Close a Staking Auction AMO

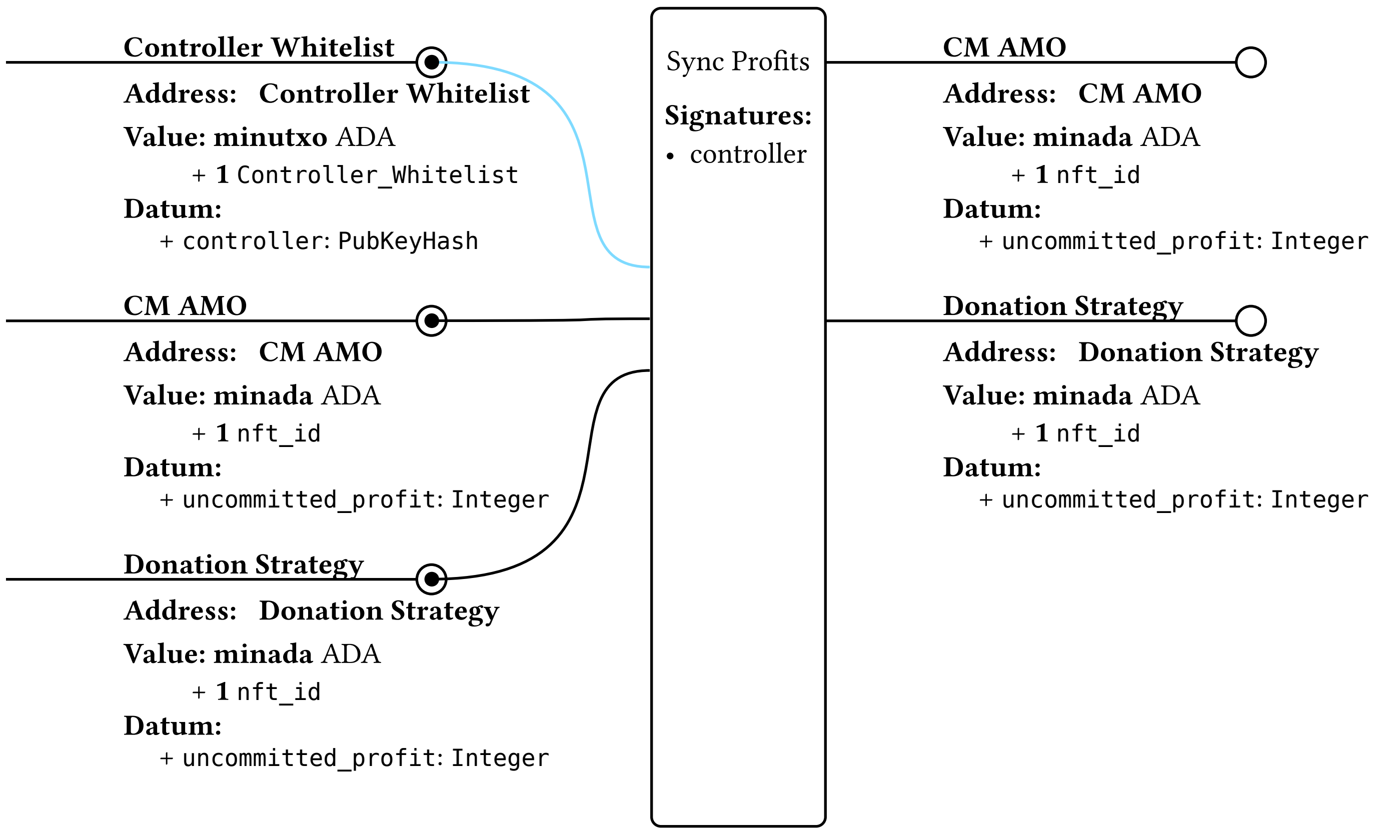


7.35. Donate via Donation Strategy

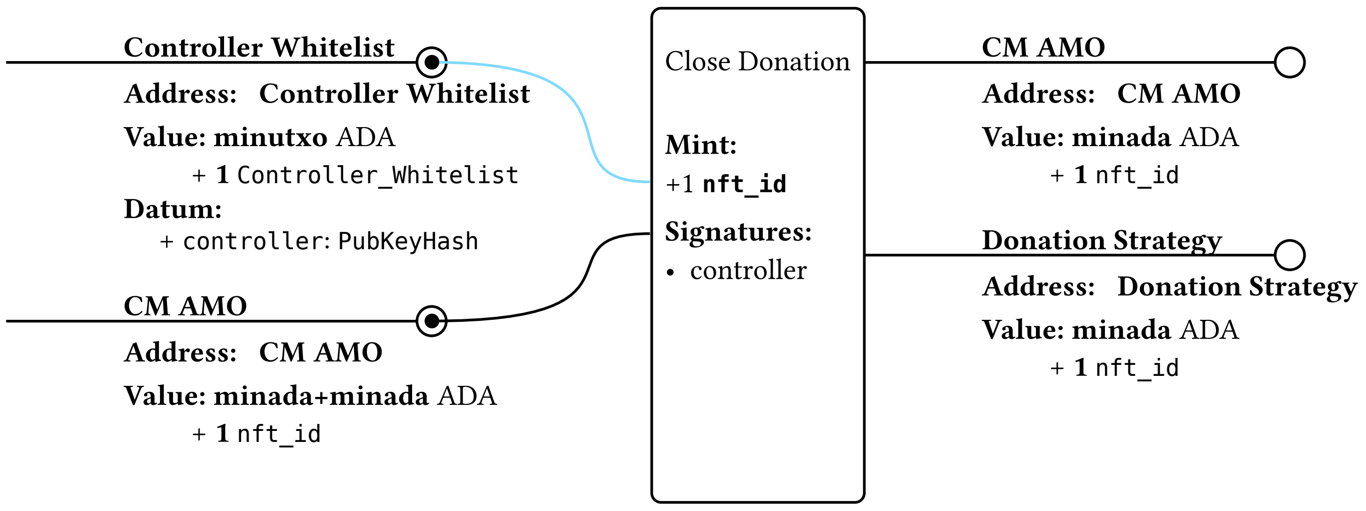


Note: uncommitted_profit += X

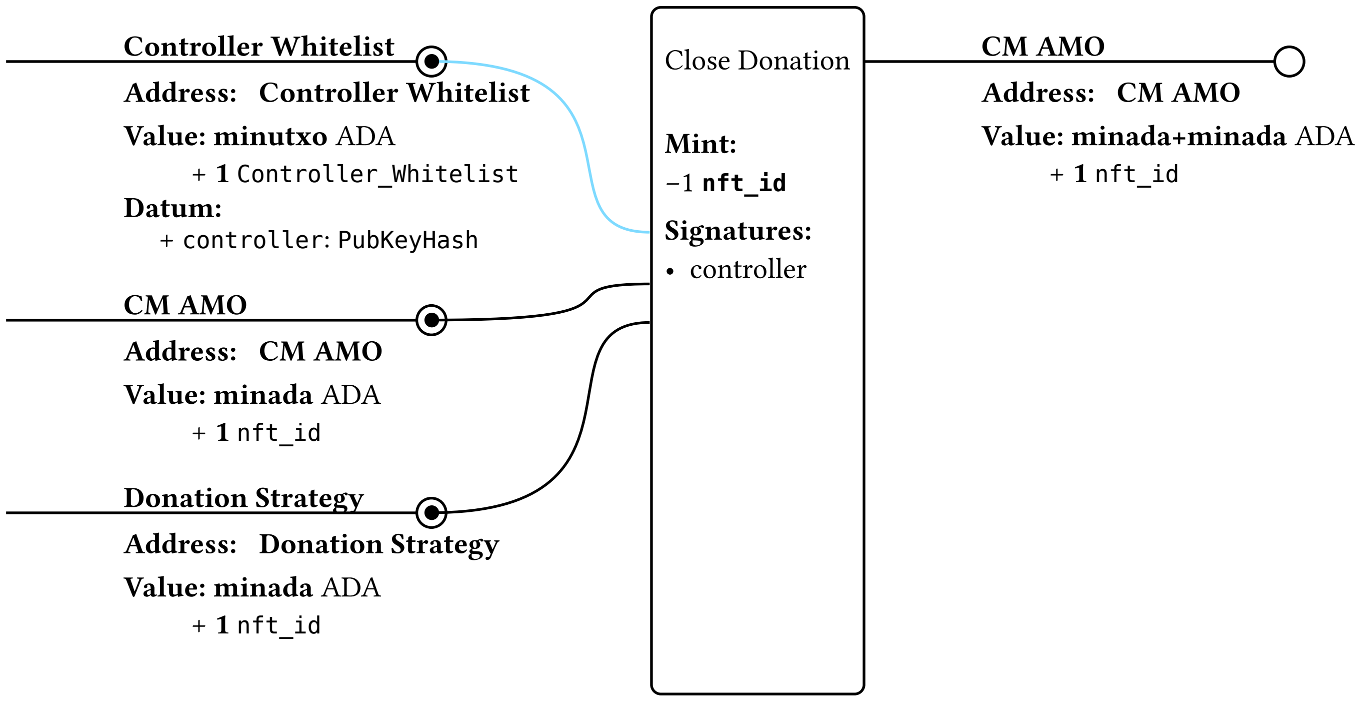
7.36. Sync Donation Profits



7.37. Open a Donation Strategy



7.38. Close a Donation Strategy



8. Deprecated Modules

No modules have been deprecated thus far

9. Changelog (System)

9.1. 1.0

- Initial Version

10. Changelog (Document)

10.1. 1.0

- Initial Version