OADA UNHCR Donation Module Technical Specification

ZYGOMEB
Optim Finance
zygomeb@gmail.com

Document Version: 2.0
System Version: 2.0
2025-05-08

For the latest version visit [THIS REPOSITORY] .

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

The 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.

1.1. System Overview

The OADA System is a decentralized financial architecture built atop the OTOKEN framework to create a fully backed ADA derivative (OADA), designed for low risk yield generation and capital efficiency. Its structure is modular, governed by the ODAO through a soul token based upgrade system, and centered on Algorithmic Market Operations (AMOs) that deploy and manage user capital through various DeFi strategies. At its core, the OADA system mints the OADA token 1:1 for ADA and allows users to stake it into sOADA to earn yield from system managed strategies. These strategies include:

- DEX AMO: Peg stabilization and liquidity provision through a stableswap pool.
- Staking Auction AMO: Dutch auction mechanism selling staking rights.
- Collateral Management AMO: The accounting hub for profit/loss across strategies.
- Staking AMO: Handles exchange rates and accounting for sOADA staking.
- Donation Module: Accepts direct ADA donations as system profit.
- Controller Whitelist: Ensures only authorized governance actors can invoke AMO actions.

All profits flow through a central profit sink and are reflected in the sOADA/OADA exchange rate, benefiting stakers.

2. UNHCR Donation Module

The UNHCR Donation Module is a specialized extension of the OADA framework designed to facilitate recurring, yield based humanitarian contributions in partnership with Switzerland for UNHCR. Rather than requiring the donation of principal assets, the module allocates the yield generated by user deposited capital, thereby preserving the user's underlying principal. Users can easily select to donate 25%, 50%, 75%, or 100% of their earnings. This simplified approach preserves the user's principal investment while creating sustainable, long-term humanitarian funding streams.

Functionally, the module builds on the same underlying architecture as the Donation Strategy (Section 6.8) but incorporates additional mechanisms to support automated, transparent, and consistently higher impact charitable disbursements. By interfacing with OADA's Collateral Management AMO, the donation module ensures precise tracking of contributed yield, provides verifiable on-chain proofs of donation, and integrates seamlessly with the broader suite of OADA strategies for multi strategy yield generation.

2.1. UNHCR Donation Module in System Context

The UNHCR Donation Module is a specialized extension that is modular and works with the OADA architecture via the OADA datum structure from an accounting standpoint to track yield and extract the appropriate amount for donations.

It is not a core component of the OADA system and sits on top of its architecture as an extension. Its purpose is to enable non custodial, yield only donations to humanitarian efforts, specifically via Switzerland for UNHCR.The interconnections between the core OADA system and the UNHCR Donation Module is primarily the compatible datum structure that enables sOADA, the OADA system strategy vault that generates yield, to interface with the scripts that comprise the Donation Module and NFT Minting Module

It introduces the following key features:

- Automated Yield Donation: Users deposit ADA (and eventually OUSD), and select a yield donation rate (25–100%). The principal remains untouched; only generated yield is routed to UNHCR.
- NFT Impact Certificates: Minted on unstake to provide on-chain, verifiable proof of cumulative donations using CIP 68 to embed yield/donation data into NFT metadata.
- Integration: Leverages existing OADA strategies (staking, auctions, LPing) to generate yield, but is simply a vault compatible with the asset datum
- Governance Aligned Disbursement: Yield is transferred to UNHCR under preset schedules with full transparency via on-chain proofs and auditability.
- Extensibility: Designed for future support of new tokens, donation recipients, and third party humanitarian partners through modular design.

Within the OADA system, the Donation Module acts as a purpose specific frontend and backend flow that reuses existing system mechanics (staking, yield harvesting, profit routing) while directing a user defined share of the returns toward a designated humanitarian sink. It introduces no systemic risk, as it does not interfere with capital solvency or the OADA peg, and is audited through the same profit accounting and Controller enforcement as all other AMOs.

2.2. Automated Yield Donation Protocol

The Automated Yield Donation Protocol provides a mechanism for users to deposit their assets—currently ADA, with future support planned for OUSD and other yield-bearing tokens—into specialized donation vaults. Once deposited, the vaults employ OADA's multi-strategy yield generation (including stake delegation, stake auctions, and liquidity provision), all governed by risk constraints defined within the Collateral Management AMO.

At configurable intervals (e.g., each epoch or on a fixed-day cycle), a user-specified percentage of the realized yield is rerouted to Switzerland for UNHCR. Crucially, the depositors' principal remains unaffected and remains at their disposal, minimizing risk while enabling meaningful, recurring donations. By decoupling yield from capital, the protocol ensures donors can maintain full ownership of their underlying assets while sustainably contributing to humanitarian initiatives.

Yield Donation System follows these steps:

1. When depositing, the user selects donation percentage (25%, 50%, 75%, or 100% of yield)
2. This selection is stored with their deposit record. The calculation is as follows:

soada_yield = (soada_exchange_rate_at_close - soada_exhange_rate_at_open) x 0.999 x soada_in_donation_box

donation_rate = soada_yield x selected_donation_percentage

donation_amount = donation_rate / (0.999 x soada_exchange_rate_at_close x soada_in_donation_box)

2.3. NFT Impact Certificate

When a user initiates an unstake, the system mints an on-chain Impact Certificate NFT as a permanent record of the user's humanitarian contributions. This NFT tracks the amount donated after the unstake transaction is processed, enabling verifiable on-chain proof of philanthropic activity by embedding metadata and cross referencing OADA's Staking AMO.

Beyond serving as a transparent ledger of charitable giving, the Impact Certificate also acts as a gateway to community based recognition. Depending on configurations set by Switzerland for UNHCR and other partners, the NFT may grant holders access to special events, exclusive donor programs, or other engagement opportunities, enhancing donor visibility and fostering an ecosystem where impactful contributions are both validated and meaningfully rewarded.

2.4. Integration with the OADA Ecosystem

The UNHCR Donation Module seamlessly integrates with OADA's existing multi strategy yield framework, thereby inheriting a robust suite of stake delegation, stake auction, and liquidity provision capabilities. Underpinning this integration is the Staking AMO, which ensures precise accounting of deposited assets, yield generation, and donation disbursements across the system.

In keeping with OADA's general design, the module's automation logic, such as epoch based yield calculations and donation schedules, relies on the same Controller Whitelist mechanism that orchestrates other AMOs, minimizing operational overhead while maintaining consistent governance standards.

Over time, the module's scope will be expanded to incorporate OUSD for dollar pegged donations and additional yield sources, accommodating a broader range of user preferences and enhancing the overall flexibility of the donation ecosystem.

2.5. Humanitarian Partnership

Switzerland for UNHCR serves as the steward of all donated yields, overseeing their timely application toward priority humanitarian initiatives. This partner driven model ensures that funds are strategically directed to areas of greatest need while maintaining transparent and accountable recordkeeping. Periodic performance evaluations and impact assessments, conducted in tandem with on-chain data, provide both donors and ODAO participants with continuous insight into the effectiveness of the allocated capital.

2.6. Future Extensions

Roadmap enhancements to the UNHCR Donation Module include support for an expanded range of yield bearing assets (stablecoins, new OTokens) and more granular user controls for setting donation percentages. This increased flexibility will enable both larger institutional participants and casual donors to tailor their philanthropic engagements precisely to their risk and yield expectations.

Further, the open source and composable nature of the OADA architecture ensures that any new feature sets can be seamlessly integrated and adopted by other humanitarian or social impact initiatives on Cardano. By providing well documented smart contracts and a modular design, the module's functionality can be replicated or adapted for diverse funding campaigns, accelerating innovation and broadening the scope of positive real world impact throughout the ecosystem.

2.7. Key Points

- Yield Donation

  By donating only the accrued yield rather than the principal, contributors maintain asset ownership and benefit from ongoing yield generation. This model fosters sustained giving without requiring a permanent forfeiture of funds.

- NFT Based Transparency

  Impact Certificates track each user's cumulative donations on-chain, ensuring verifiable, tamper evident records. This approach bridges the transparency of decentralized finance with the real world accountability crucial for humanitarian relief.

- Scalable & Extensible

  The module is architected to support multiple yield sources and assets. Near term plans include integrating OUSD for stable, dollar denominated contributions and broadening the scope of potential liquidity farming strategies.

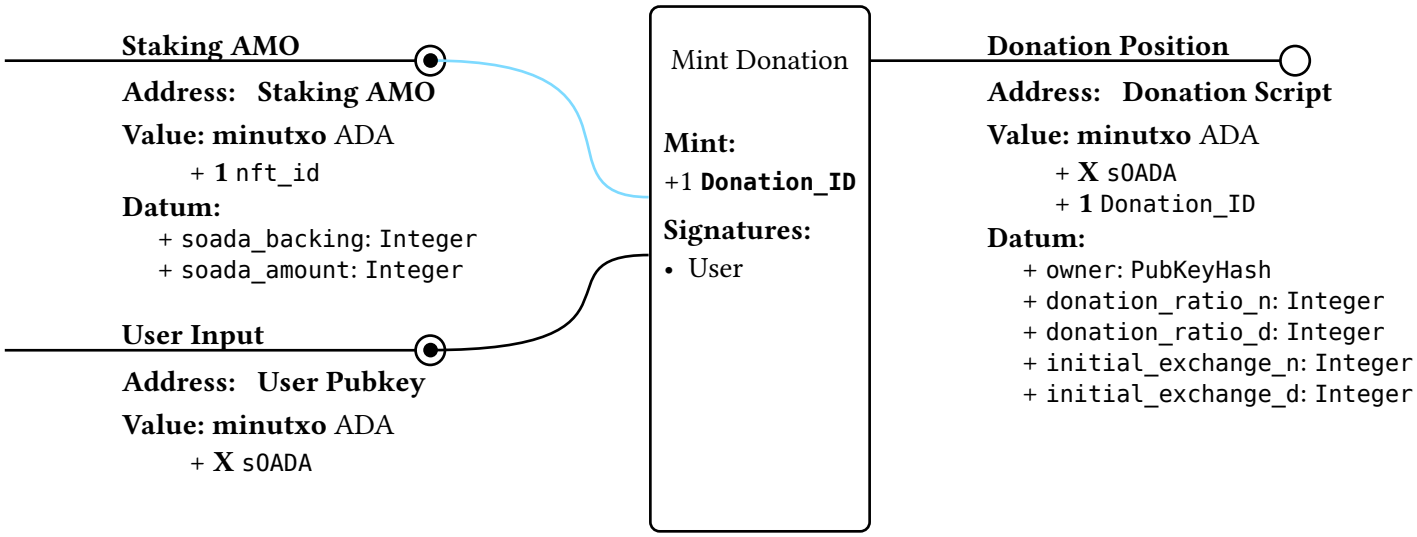
- Operational Oversight

  Switzerland for UNHCR coordinates the allocation of donated yields in pursuit of prioritized humanitarian needs. Through quarterly performance reviews and detailed impact reporting, they ensure that every donation is effectively and transparently deployed.

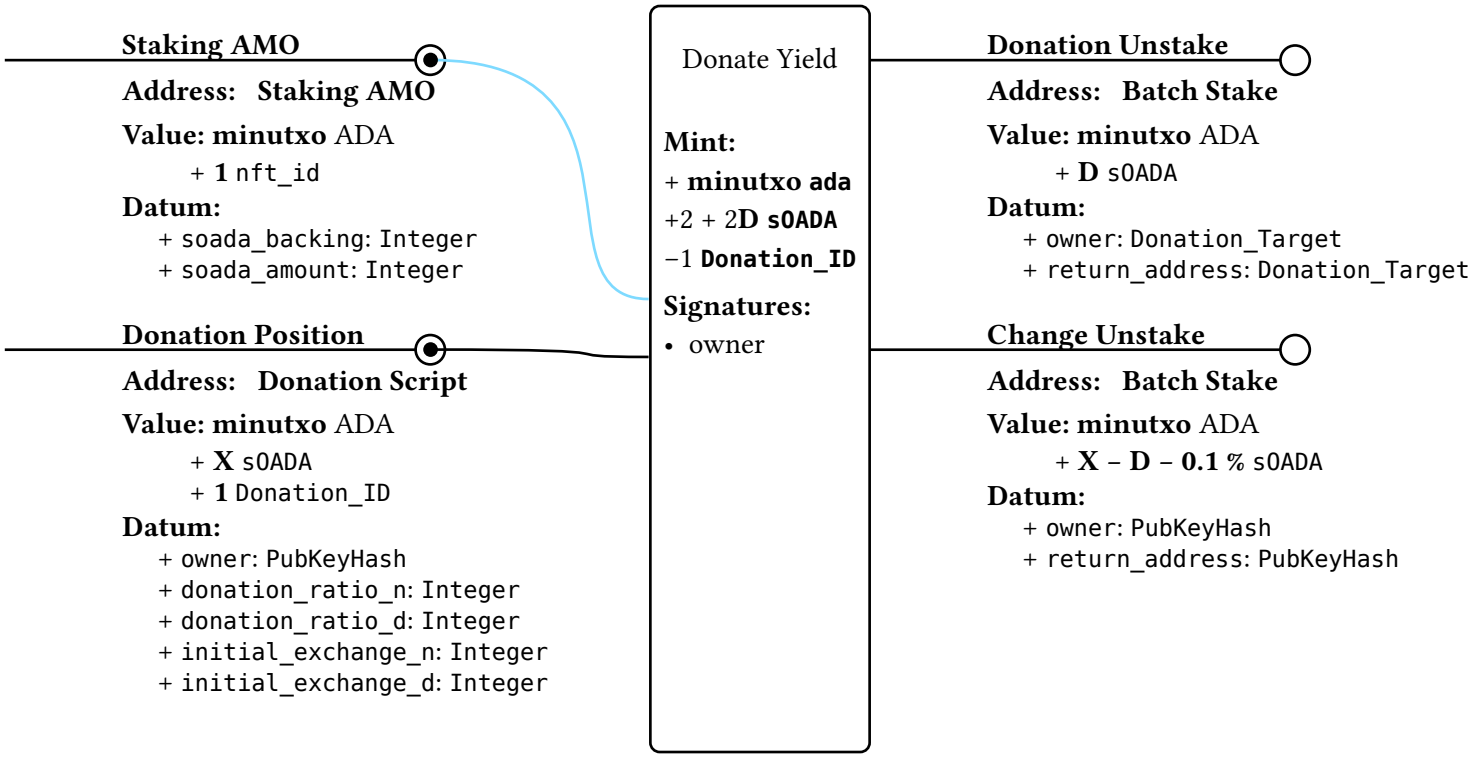
With the UNHCR Donation Module the OADA framework demonstrates how decentralized finance can continually fund critical humanitarian work in a trust minimized environment, furthering Cardano's global mission of inclusive, real world utility. By preserving the depositor's principal and donating only accrued yield, this module underscores a new paradigm in philanthropy where impactful contributions align with the long term preservation of user assets.

3. sOADA Donation Flow

3.1. 1 · Open Donation Position

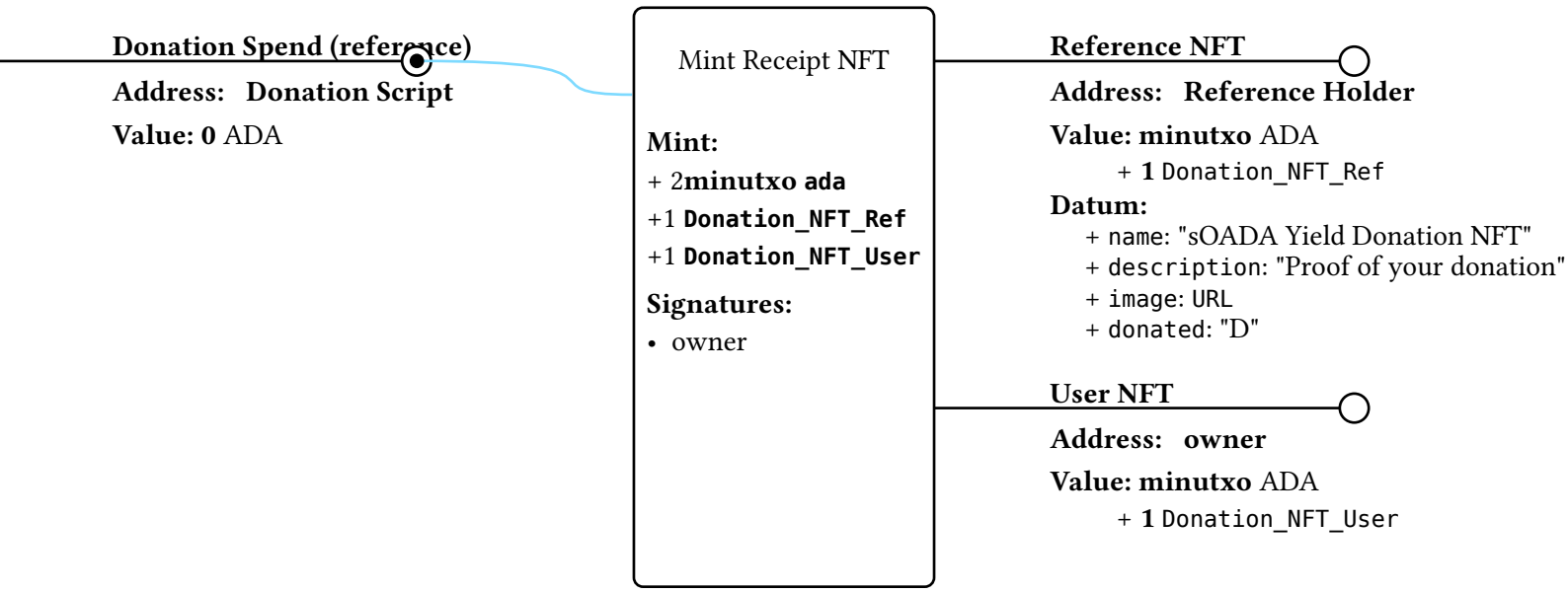


3.2. 2 · Donate Yield & Create Un-stake Orders



Note:
0.1 % of sOADA is burned (protocol fee) before splitting;
D = donation_ratio × (current − initial exchange) × X / current_exchange;
Donation_ID is burned in this transaction

3.3. 3 · Mint Donation Receipt NFT (optional, same tx as step 2)



Note: Both NFTs are minted under the donation-NFT policy; the 'Ref' copy is locked by Reference Holder.

4. sOADA Donation API

This section specifies a minimal HTTP interface that lets a wallet or backend service drive the sOADA Donation workflow without holding private keys. Every mutating endpoint returns an unsigned transaction (raw CBOR-hex) plus the datum fragments it embeds; signing and submission are left to the caller.

4.1. Resource model

Resource	Description
DonationPosition	UTXO locked by the **donate-sOADA** script holding sOADA, the single-use ID NFT and an inline datum.
UnstakeOrder	UTXO locked by the **Batch-Stake** script produced when a donation position is redeemed.
ReferenceNFT	CIP-68 reference NFT whose datum stores the exact donated amount; minted together with a user NFT.

4.2. REST interface

4.2.1. POST `/v1/donations`

Creates a **DonationPosition** and mints the ID NFT.

```
{
  "owner_vkey_hash": "string",
  "soada_amount": 123456789,
  "donation_ratio_numer": 3,
  "donation_ratio_denom": 4
}
```

⇢ 201 Created

```
{
  "tx_cbor": "hex-encoded-cbor",
  "donation_position": {
    "tx_hash": "…",
    "index": 0,
    "value": { "sOADA": 123456789, "Donation_ID": 1 },
    "datum": {
      "owner": "owner_vkey_hash",
      "donation_ratio": [3,4],
      "initial_exchange": [9000000000001, 8000000000001]
    }
  }
}
```

4.2.2. POST /v1/donations/{tx_hash}/{index}/redeem?mint_nft=

Consumes the position, splits the sOADA and (optionally) mints the NFT pair.

```
{
  "donation_target_key_hash": "string"
}
```

⇢ 201 Created

```
{
  "tx_cbor": "hex-encoded-cbor",
  "donated_soada": 1234,
  "outputs": {
    "donation_unstake_index": 0,
    "change_unstake_index": 1
  },
  "reference_nft": {
    "policy_id": "…",
    "asset_name": "…",
    "datum": {
      "metadata": {
        "name": "sOADA Yield Donation NFT",
        "description": "Proof of your donation",
        "image": "https://…"
      },
      "version": 1,
      "extra": 1234
    }
  }
}
```

4.2.3. GET /v1/donations/{tx_hash}/{index}

Returns a snapshot of a live position and computed deltas.

⇢ 200 OK

```
{
  "datum": { … },
  "soada_locked": 123456789,
  "current_exchange_rate": "1.187452",
  "yield_soada": 2345,
  "donatable_soada": 1758
}
```

4.3. Web Socket Channel

wss: /…/donations/updates pushes JSON messages whenever the AMO exchange rate changes (new current_exchange_rate), or a watched position is spent (include spend tx_hash & final donation).

4.4. Error codes

Code	Meaning
409 RatioOutOfBounds	Donation ratio ≤ 0 or ≥ 1
409 ExchangeRateMismatch	Initial exchange rate in request no longer matches AMO state
409 AlreadySpent	DonationPosition already consumed
404 NotFound	UTXO cannot be located or is not a DonationPosition
400 BadInput	Malformed numbers, insufficient sOADA, etc.

4.5. Notes on implementation details

- Unsigned transactions are encoded as raw CBOR (the field names shown are illustrative) so any wallet-connector can sign them verbatim.
- The server uses the same on-chain validator code to compute yield and to draft the redeemer; that prevents client-side rounding drift.
- NFT minting is kept purely optional so integrators that don't need the receipt can save fees.
- All numeric fields follow the Cardano convention of lovelace-level integers; no decimals are transmitted.
- The Web-socket stream carries only tips the client can verify locally; nothing security-critical is trusted from it.

5. 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.

5.1. 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

5.1.1. 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.

5.1.2. 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.

5.2. 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.

5.3. 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.

6. Capital Flows

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

6.1. 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.

6.1.1. 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.

6.1.1.1. 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.

6.1.1.2. 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.

6.1.2. 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.

6.1.3. 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.

6.1.4. 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.

6.1.5. 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).

6.1.6. 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.

7. 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.

7.1. 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.	

7.1.1. 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.

7.1.2. 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.

7.1.2.1. OADA Deposit Rule

7.1.2.2. OADA/sOADA Staking Rule

7.1.2.3. OADA Fee Claim Rule

7.1.2.4. OADA Splash DEX Rule

7.1.2.5. OADA Stake Auction Rule

7.2. 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.	

7.2.1. 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.	

7.2.2. 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.

7.2.2.1. OADA/sOADA Staking Rule

7.3. 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.

7.3.1. 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.	

7.4. 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.

7.4.1. 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.	

7.4.2. 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.	

7.5. 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.	

7.5.1. 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.	

7.5.2. 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.	

7.5.3. 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.	

7.6. 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.

7.6.1. 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.	

7.6.2. 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.	

7.6.3. 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.	

7.7. 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)

7.7.1. 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.

7.8. 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.

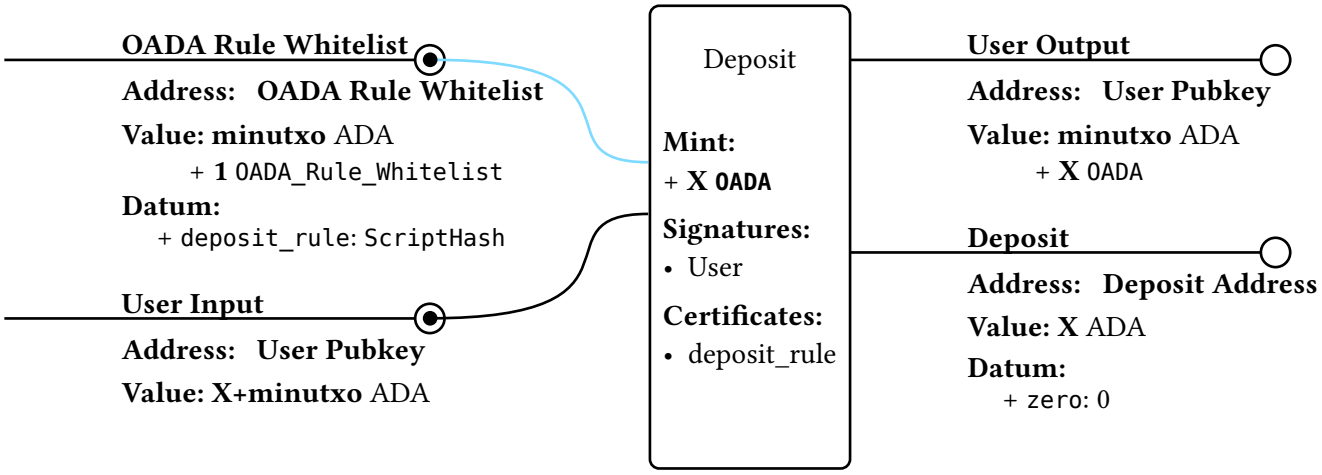
Mostly used for working around Catalyst, but also given as the simplest example of a strategy. A note on the terminology used, as it doesn't have an associated Rule, it is considered a Strategy. If it were to be enriched with a Rule, to burn donated OADA (of which donations are currently unaccepted) then it would be considered an AMO.

8. System Transactions

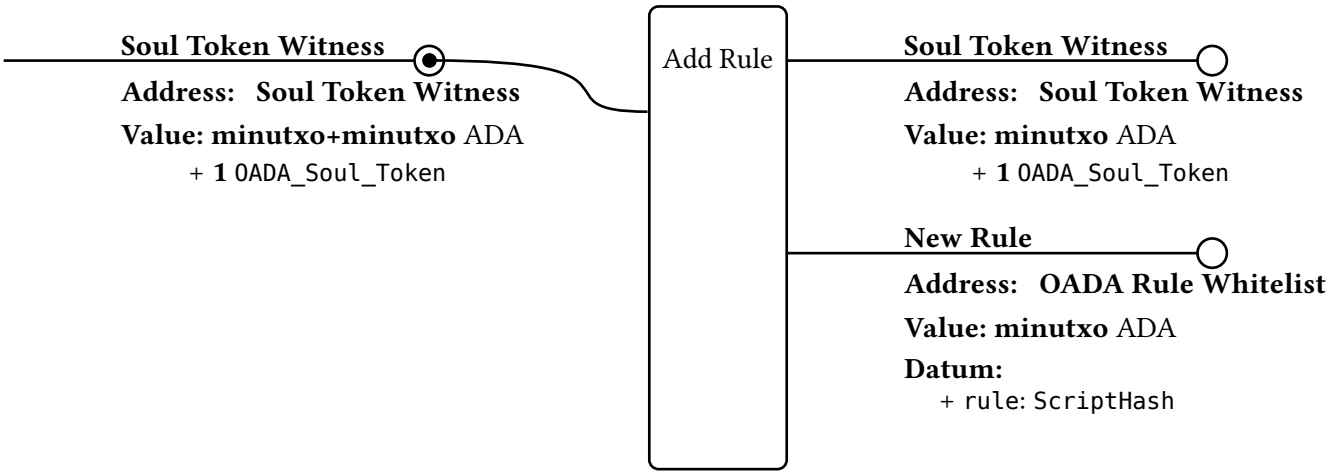
In this section we present the totality of the transaction schemas used in the system's operation. We present the *minimal* version of each transaction, as we attempted to make it possible to include other contracts and transfers while performing the system operations – nevertheless do verify in code 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.

8.1. Mint OADA

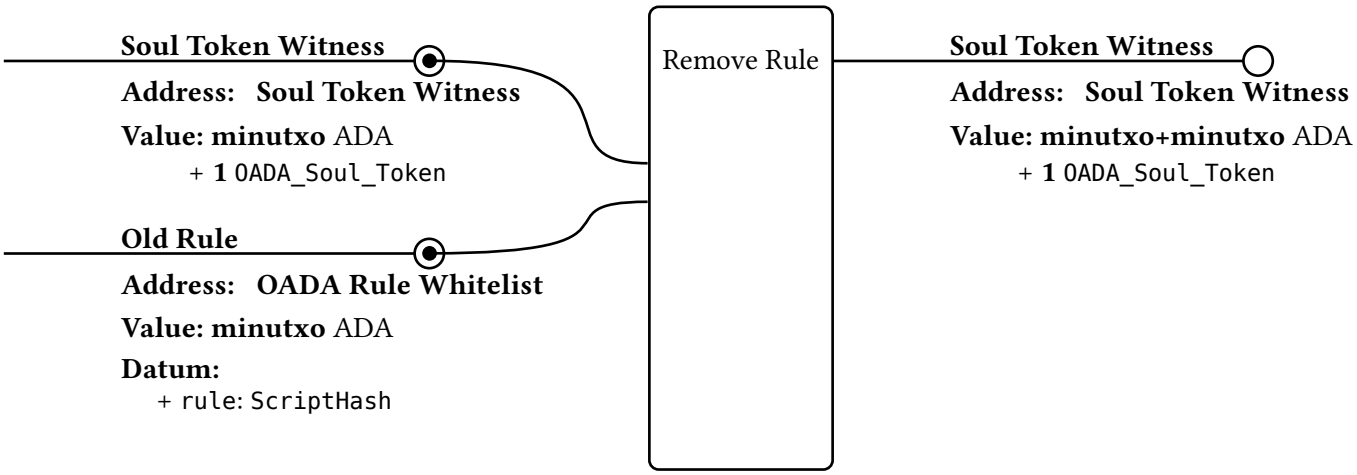


8.2. Add OADA Rule



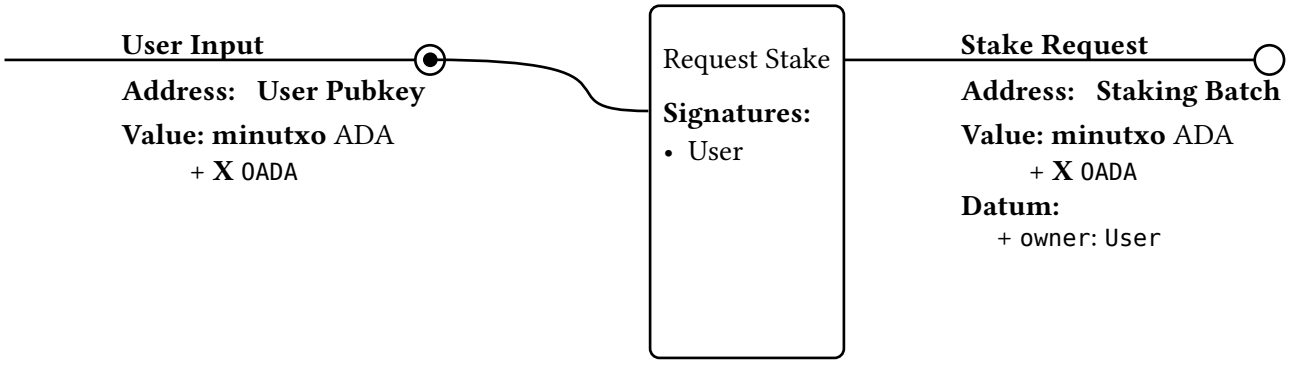
Note: Sctipt is inlined

8.3. Remove OADA Rule

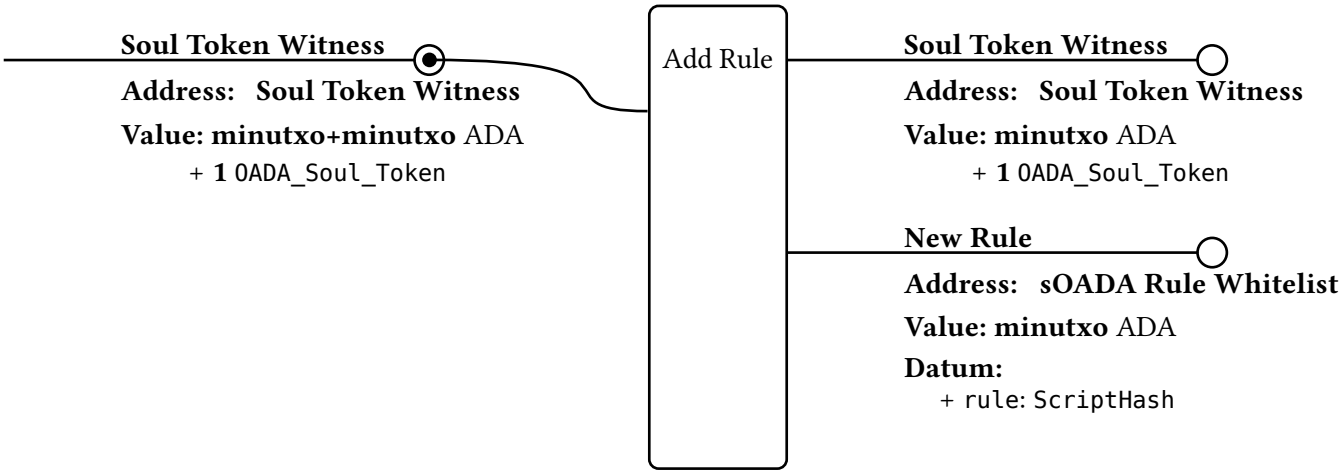


Note: Sctipt is inlined

8.4. Request Stake of OADA

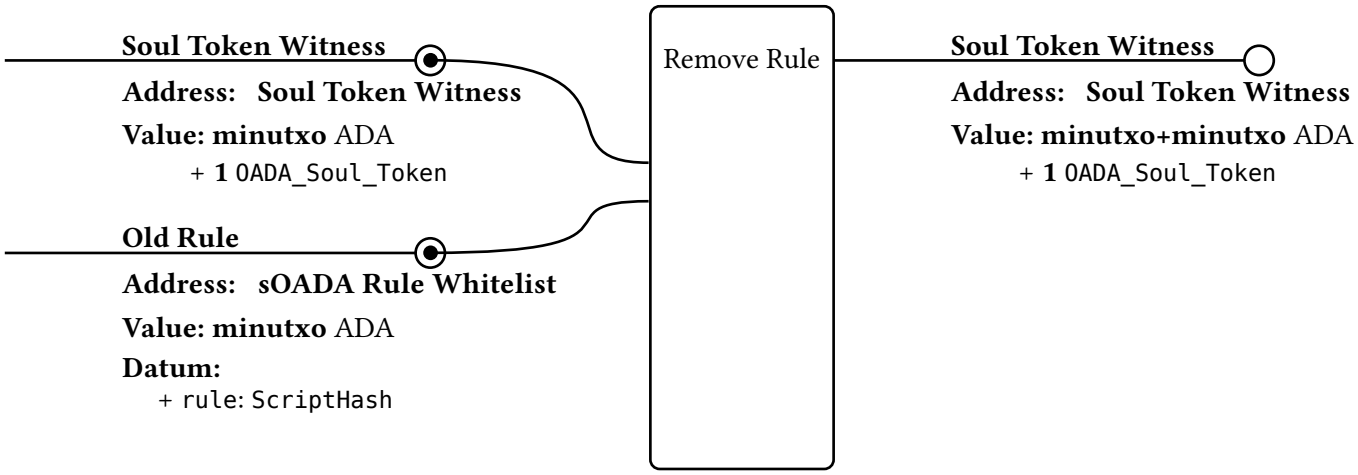


8.5. Add sOADA Rule



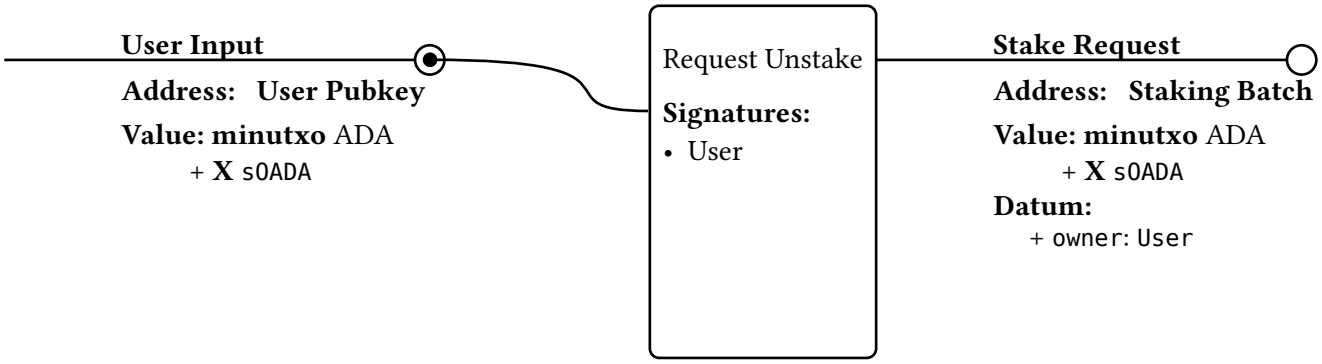
Note: Sctipt is inlined

8.6. Remove sOADA Rule

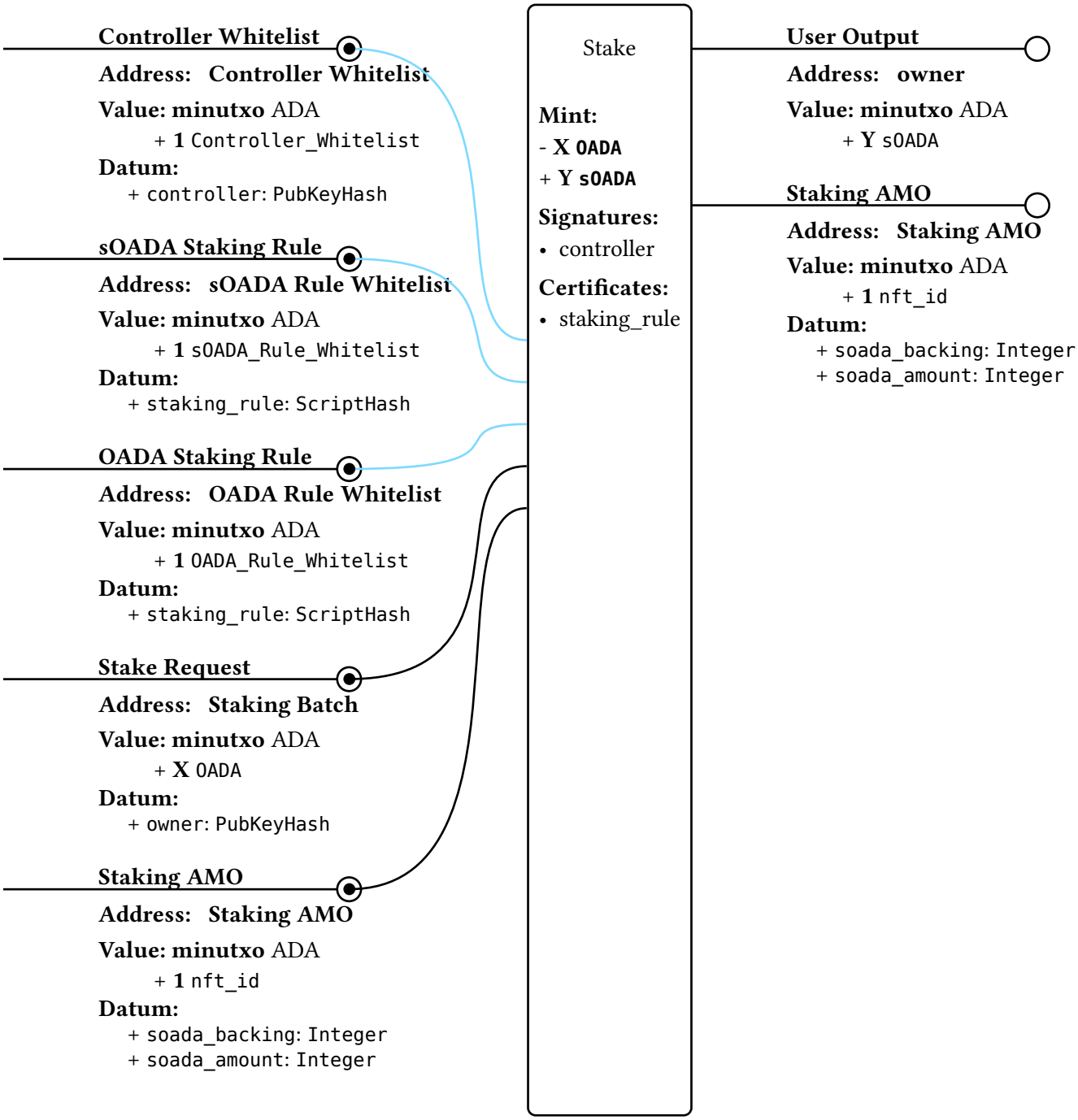


Note: Sctipt is inlined

8.7. Request Unstake of sOADA

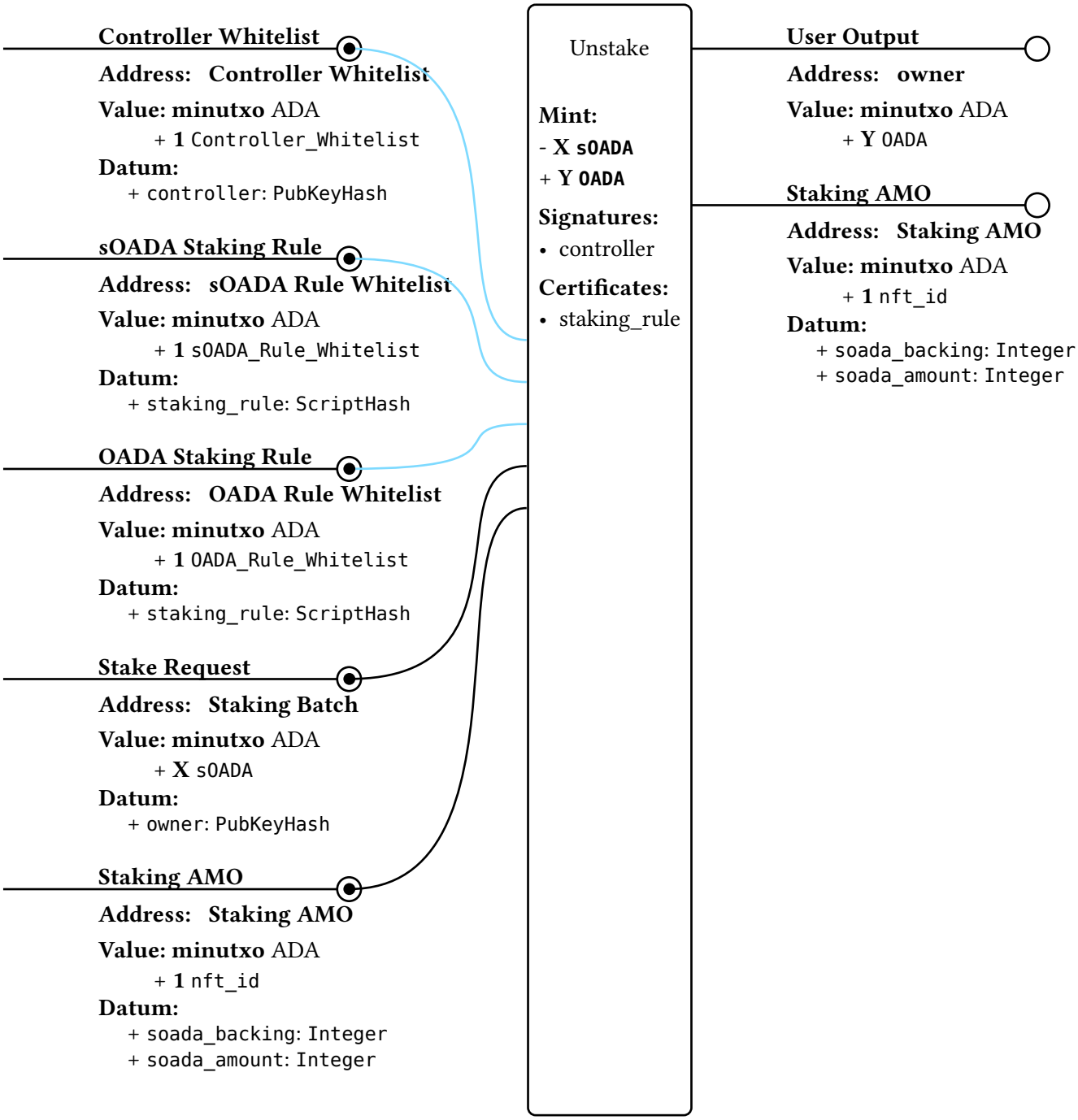


8.8. Stake OADA for sOADA



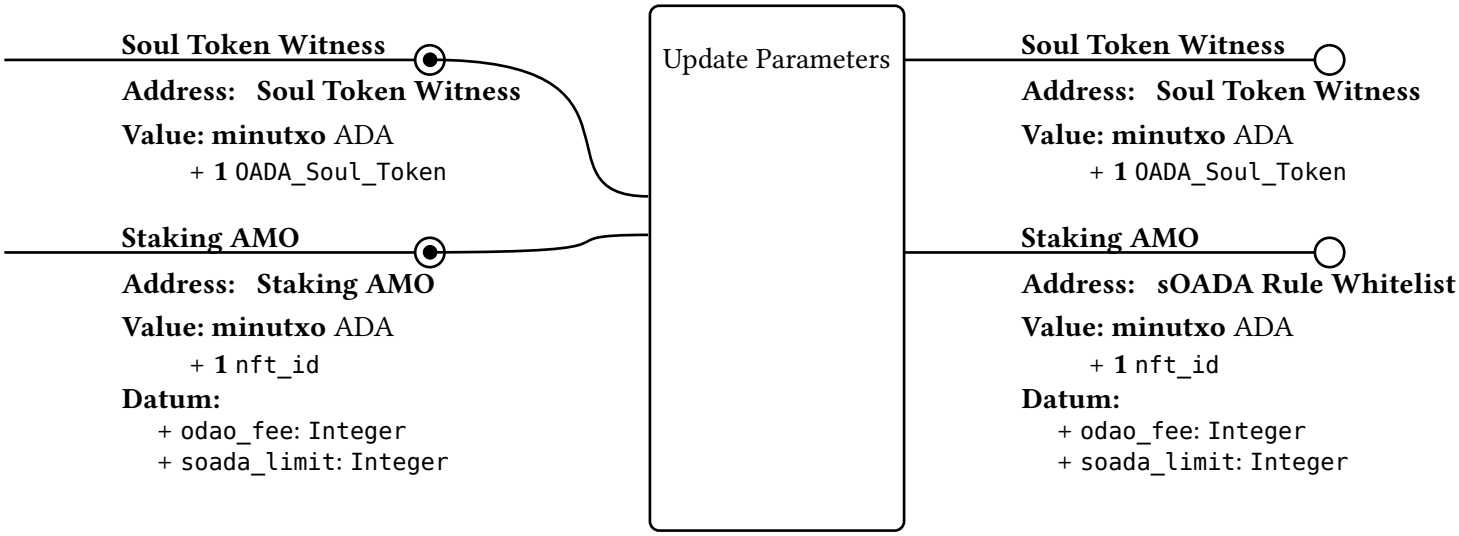
Note:
Y = X*sOADA_backing/sOADA_amount
Updates the 'soada_backing' and 'soada_amount' to reflect the action
The 'staking_rule' is the same script for OADA and sOADA

8.9. Unstake sOADA for OADA

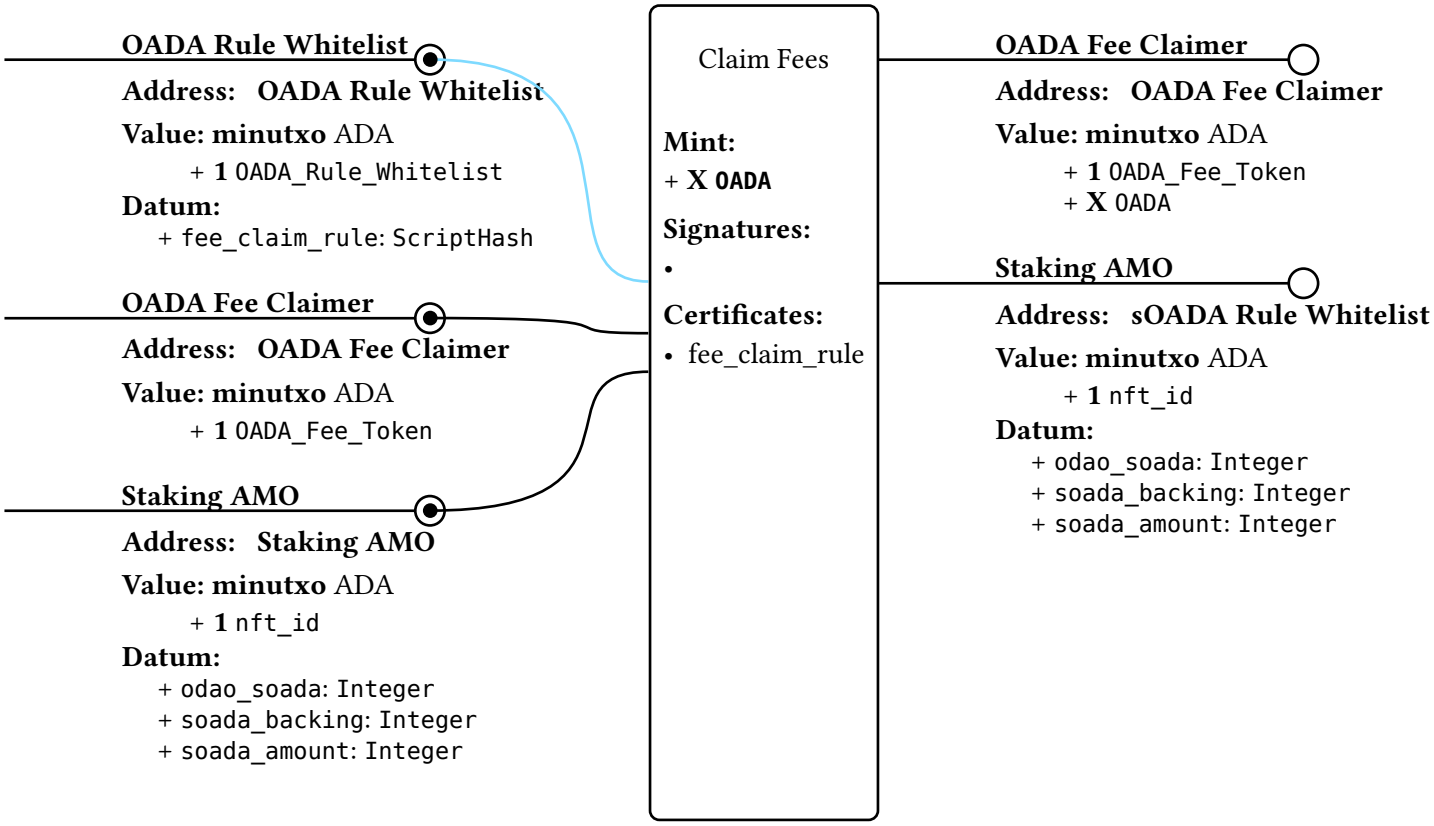


Note:
Y = X*0.999*sOADA_amount/sOADA_backing
Updates the 'soada_backing' and 'soada_amount' to reflect the action
The 'staking_rule' is the same script for OADA and sOADA

8.10. Perform Staking AMO Paramerter Update

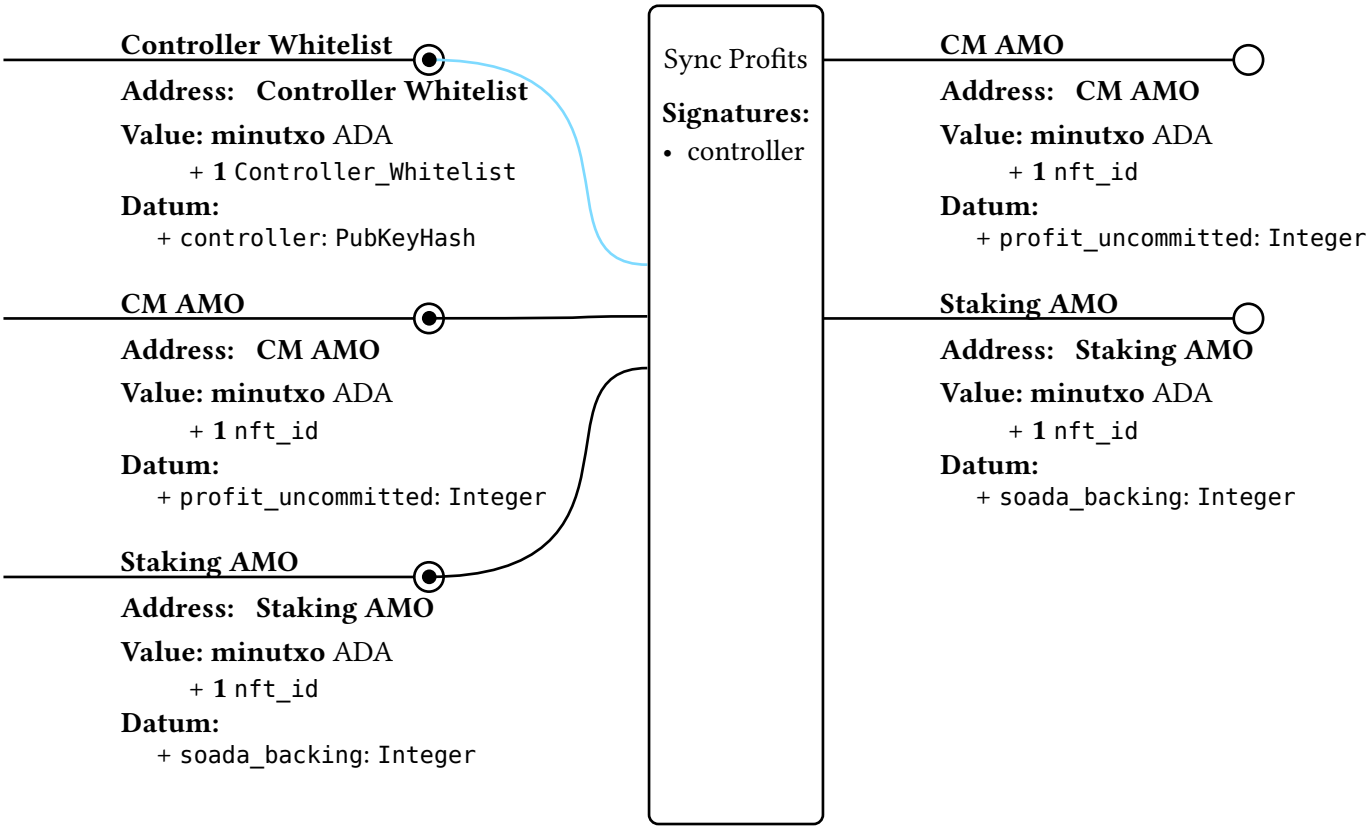


8.11. Claim ODAO Fees



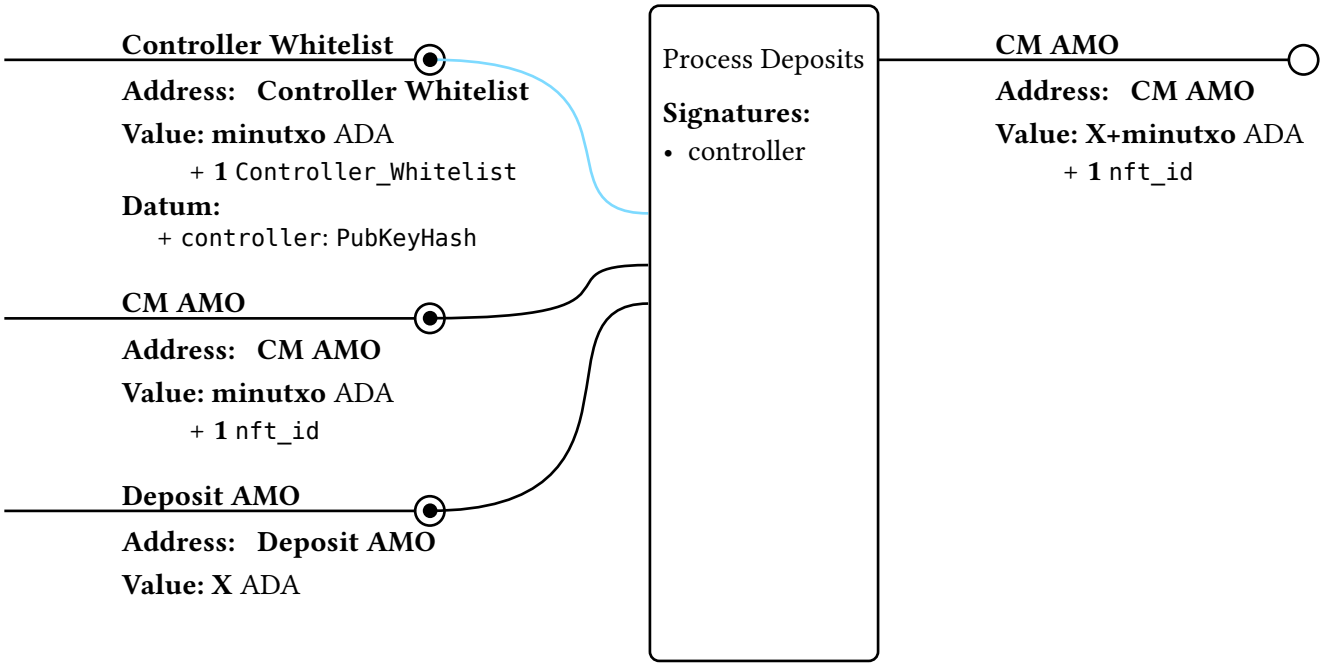
Note:
X = (input 'odao_soada' - output 'odao_soada') * soada_backing / soada_amount
Importantly, upon fee claim, the odao skips the 0.1% unstake fee present on regular staking

8.12. Sync Profits with Staking AMO



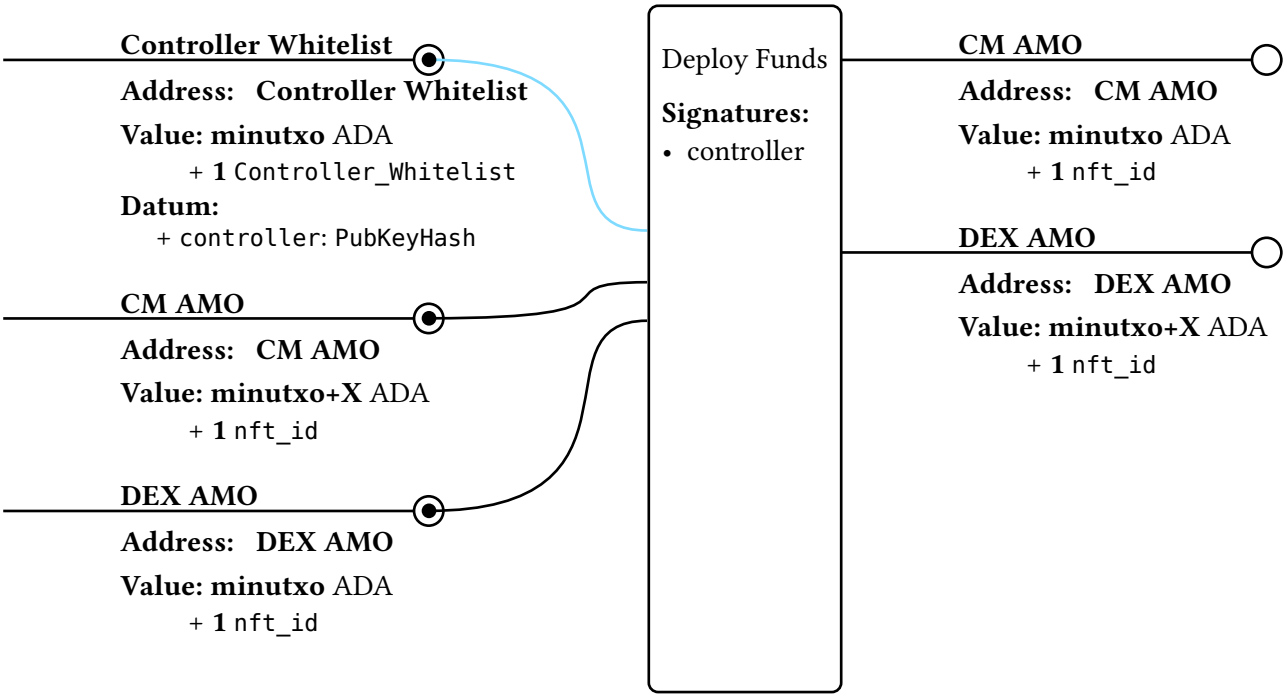
Note: profit_uncommited is added to soada_backing

8.13. Process Deposits in Collateral Management AMO

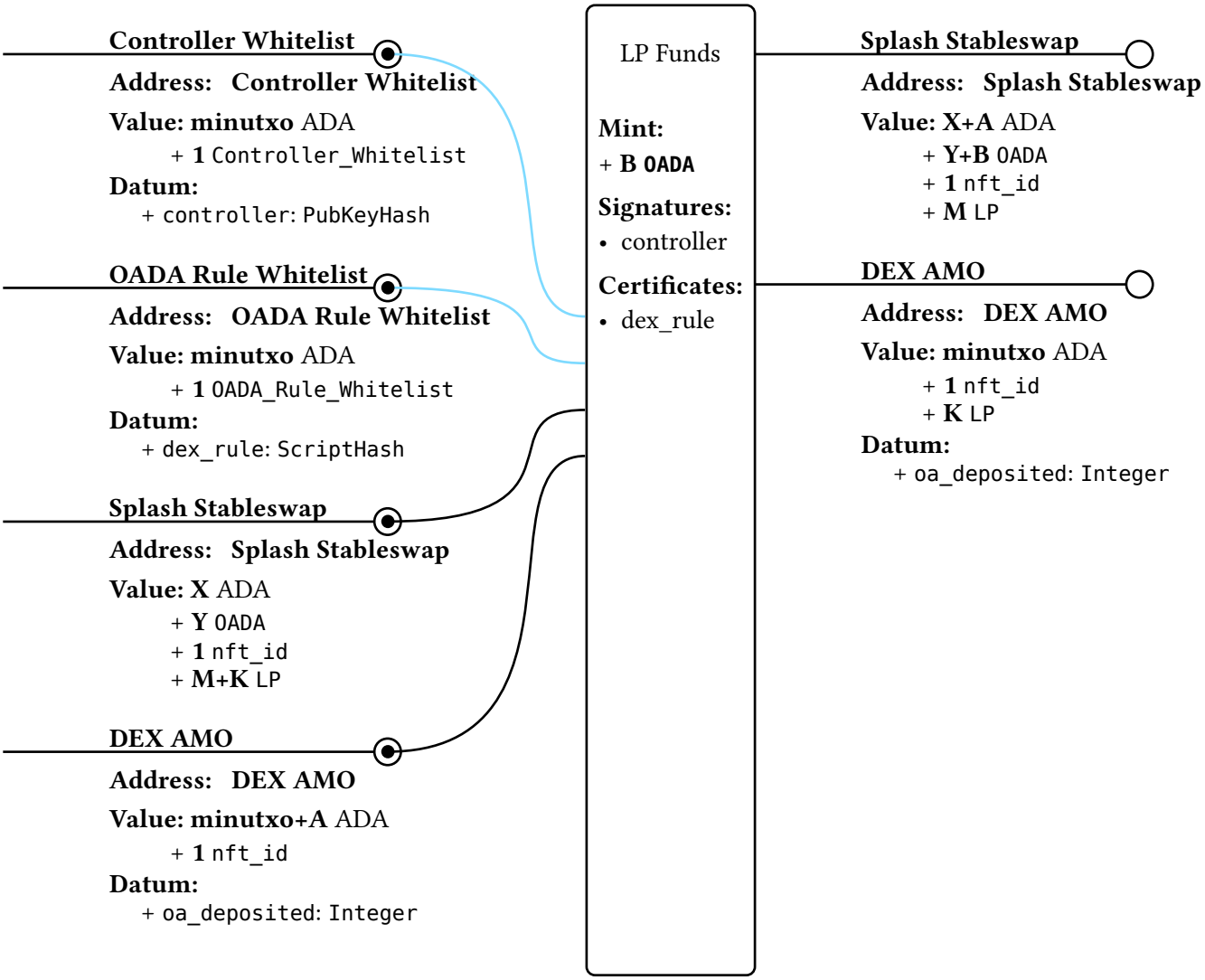


Note: Can process multiple dozen deposits at a time

8.14. Deploy Funds into the Splash DEX AMO

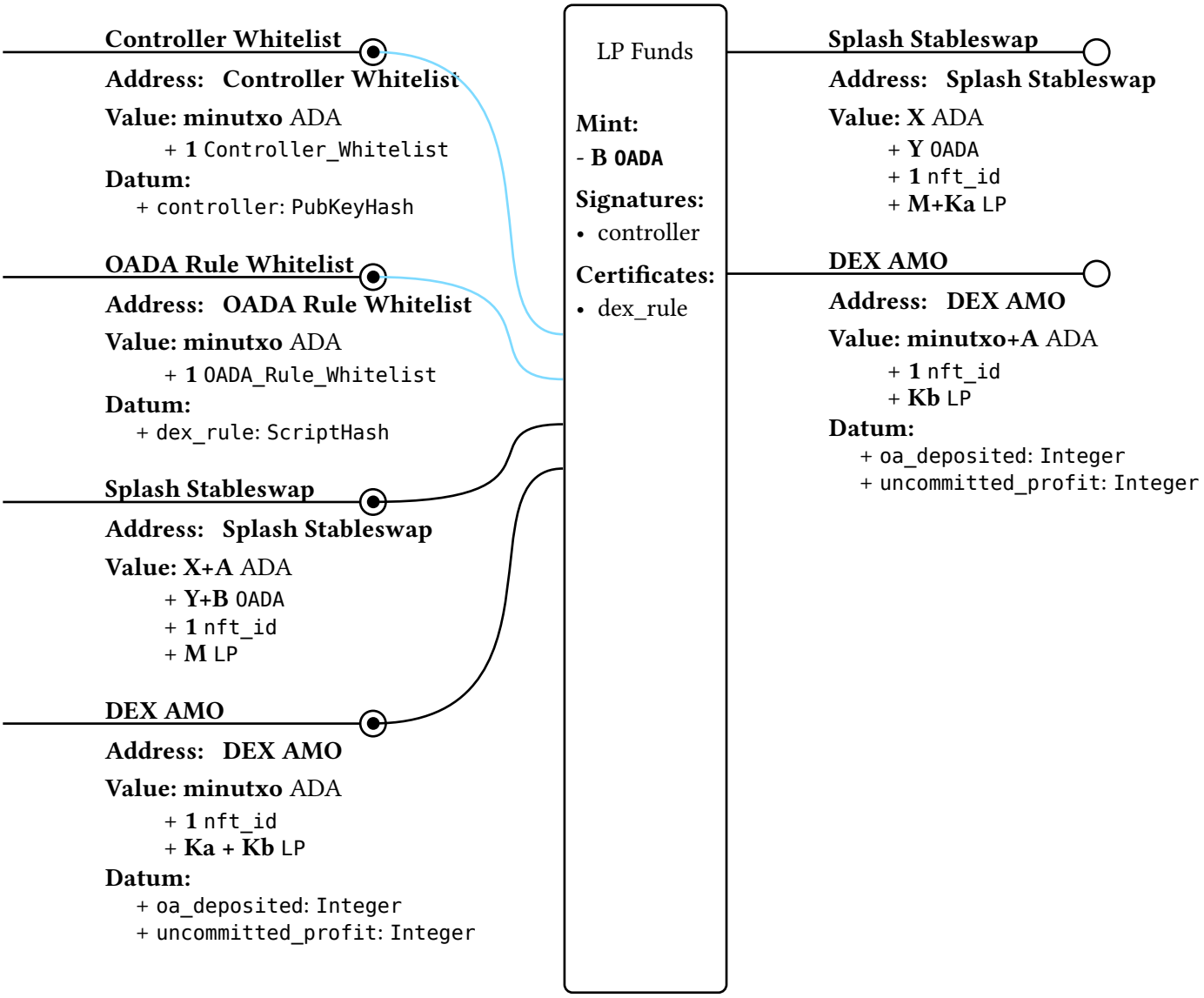


8.15. Deposit ADA with OADA into Splash Stableswap



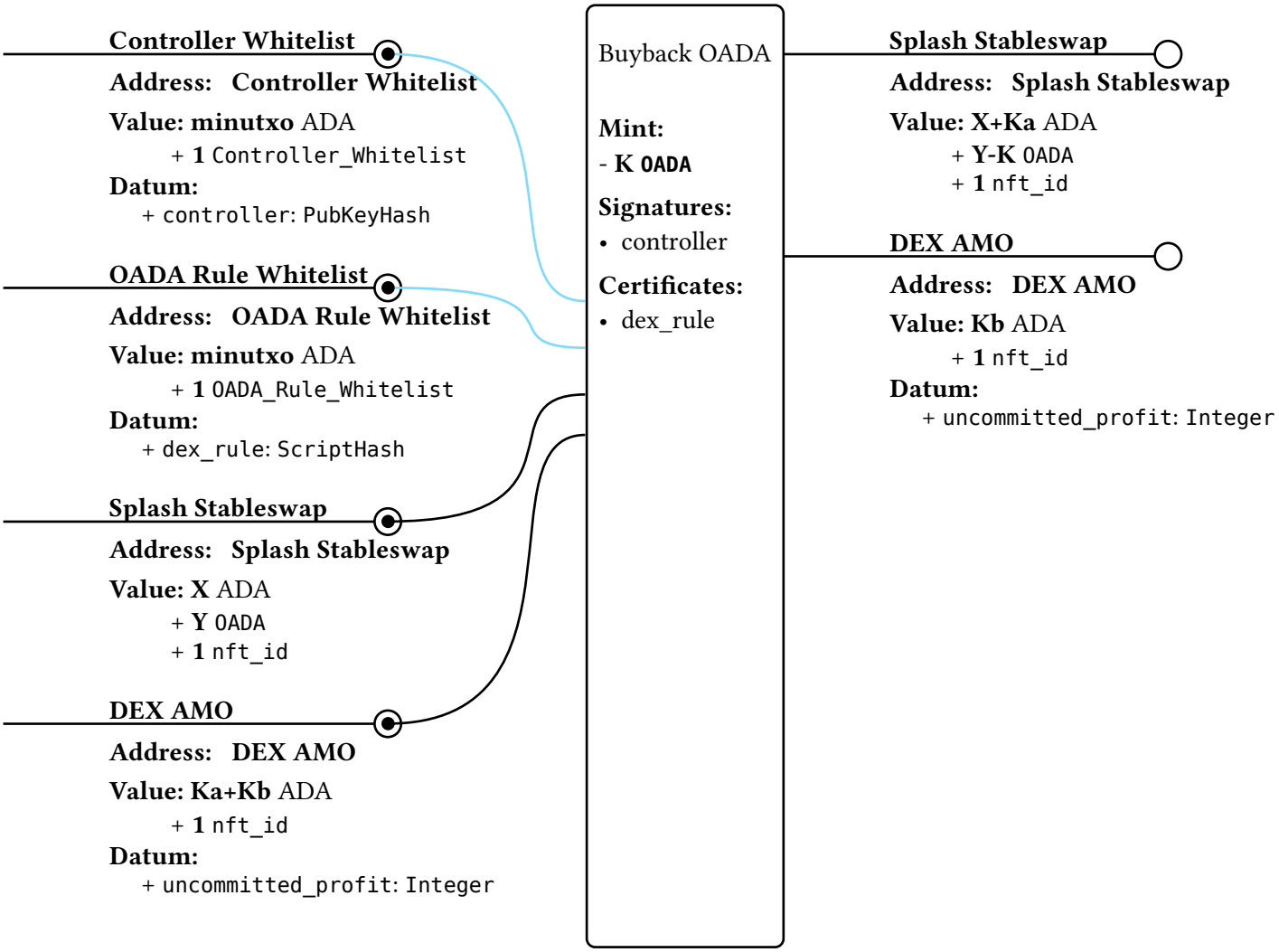
Note:
deposit_guard <= X / Y
deposit_guard >= Y / X
deposit guard ensures the deposit happens between 0.999 and 1.001
A+B is added to oa_deposited

8.16. Withdraw LP from the Stableswap



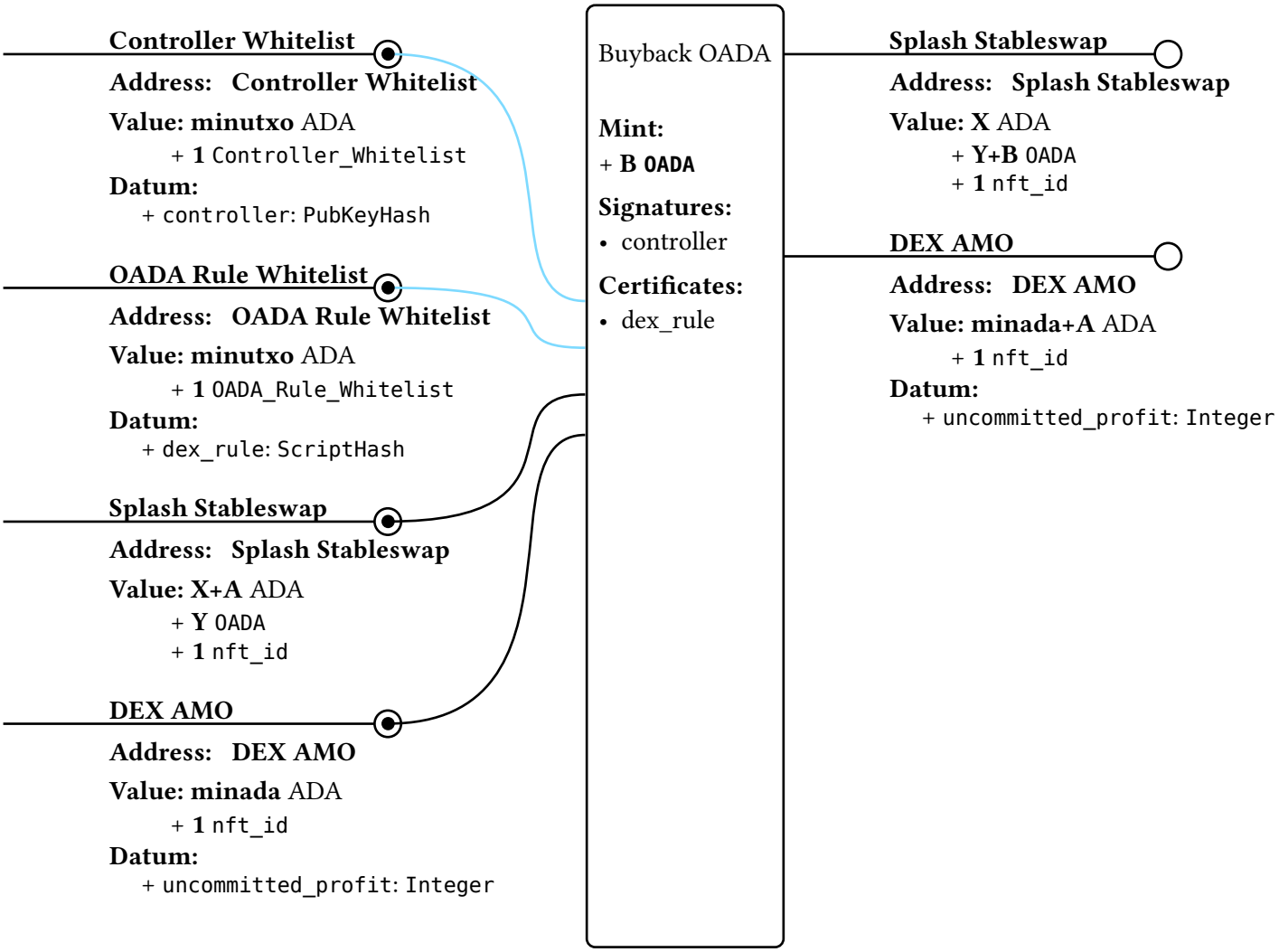
Note:
A+B is subtracted from oa_deposited
profit += (withdrawn - deposited) * (Ka / (Ka + Kb))

8.17. Protect Downside Peg on the open market



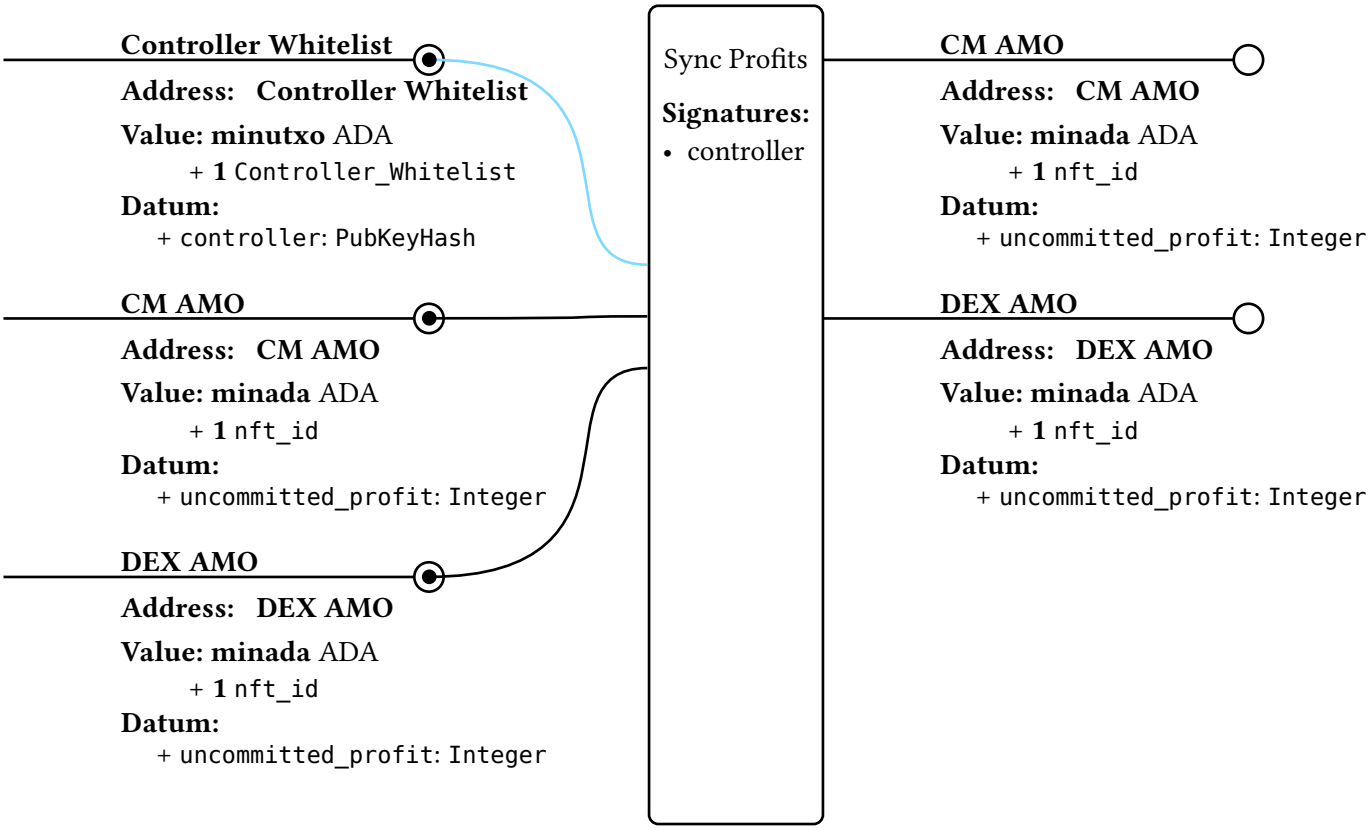
Note:
K >= Ka
profit += K - Ka

8.18. Protect Upside Peg on the open market

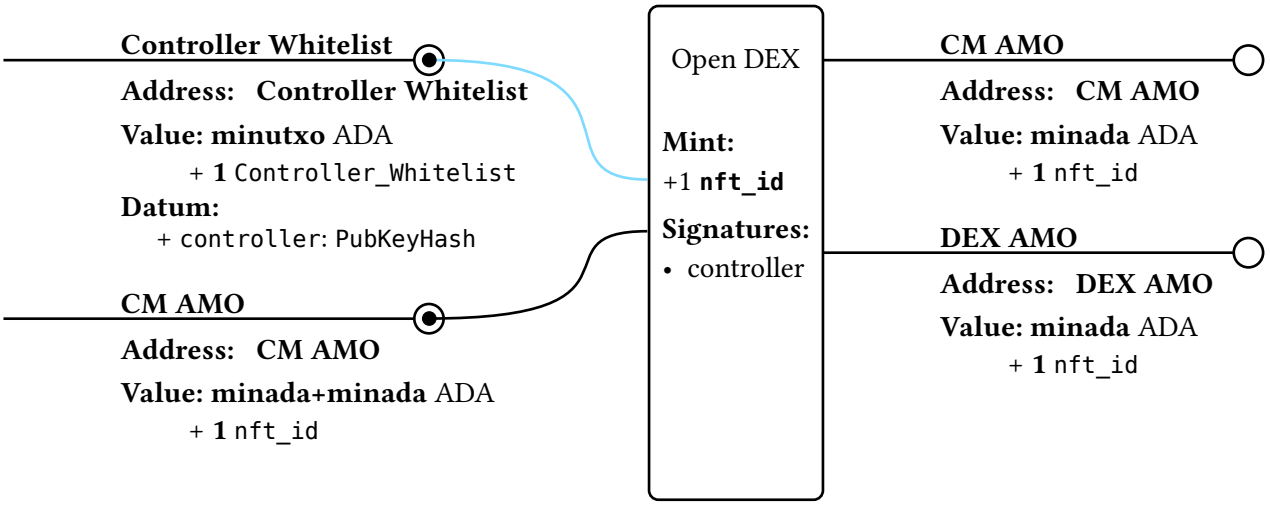


Note:
A >= B
profit += A - B

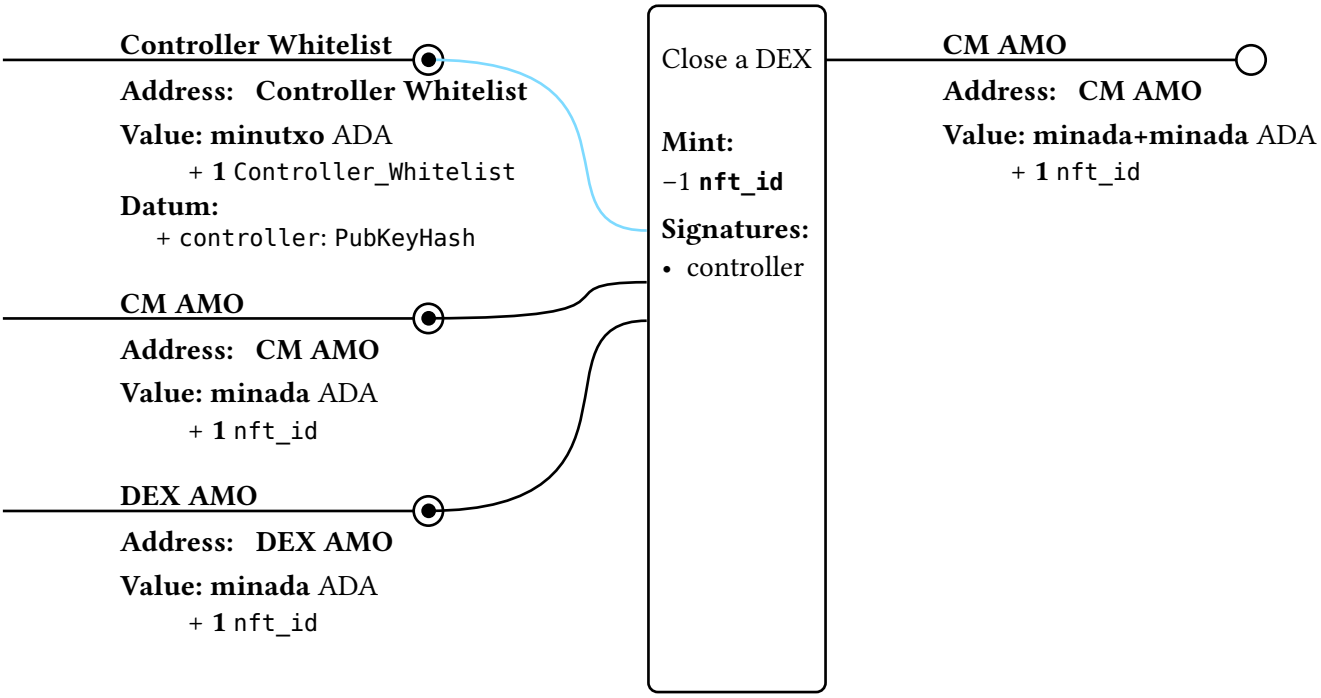
8.19. Sync DEX AMO Profits with the CM AMO



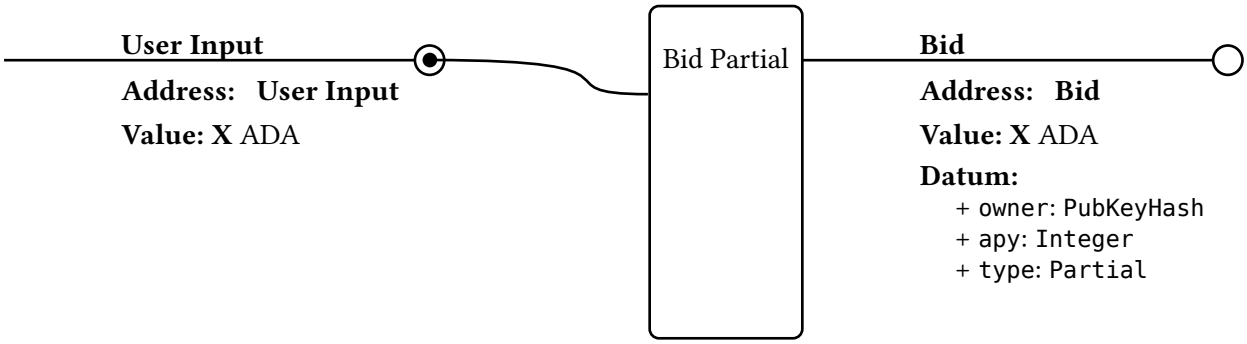
8.20. Open a DEX AMO



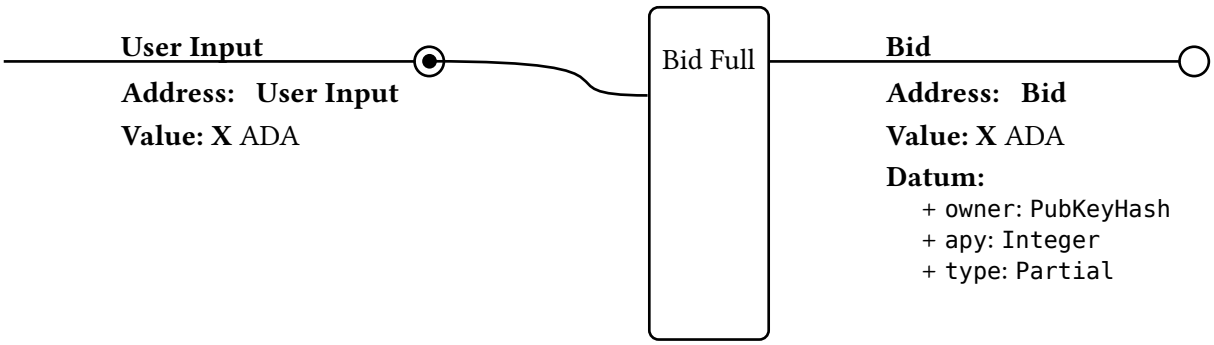
8.21. Close a DEX AMO



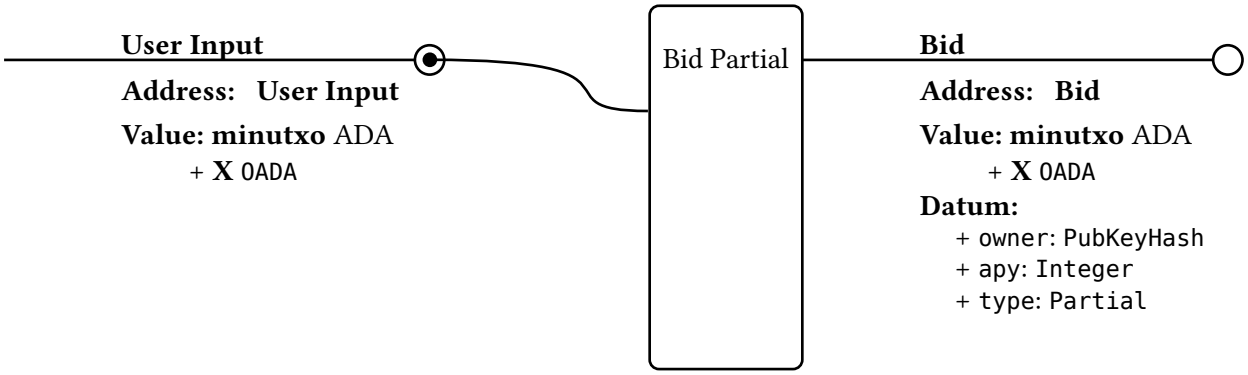
8.22. Partial Fill Bid ADA in the Staking Auction



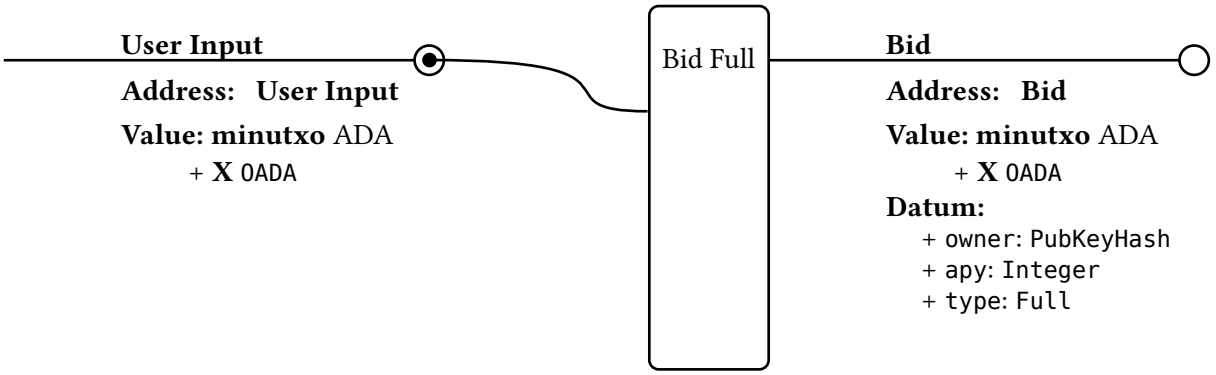
8.23. Fill or Kill Bid ADA in the Staking Auction



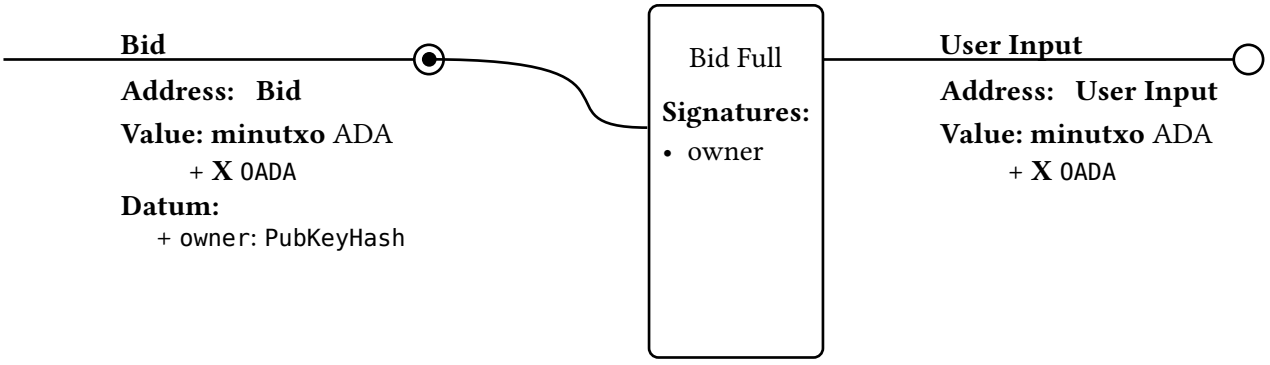
8.24. Partial Fill Bid OADA in the Staking Auction



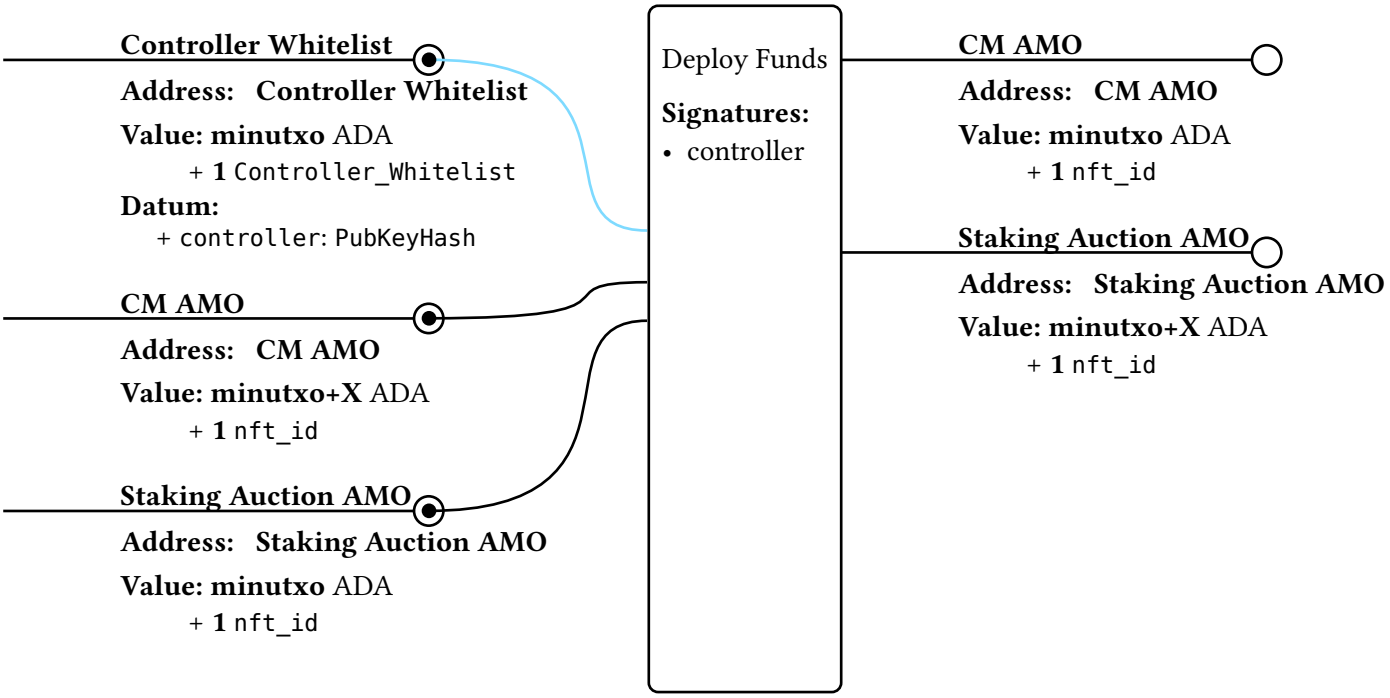
8.25. Fill or Kill Bid OADA in the Staking Auction



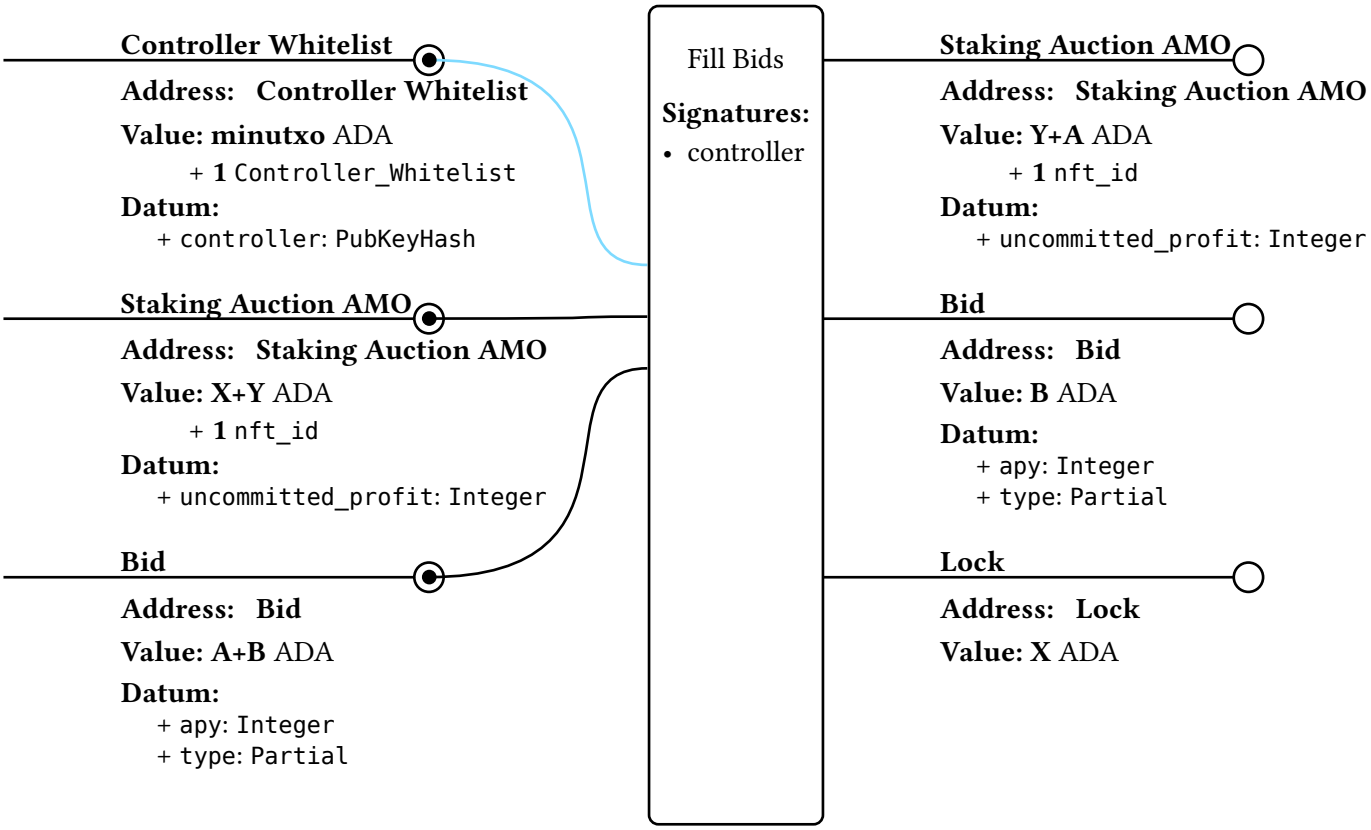
8.26. Cancel a bid



8.27. Deploy Funds into the Staking Auction AMO

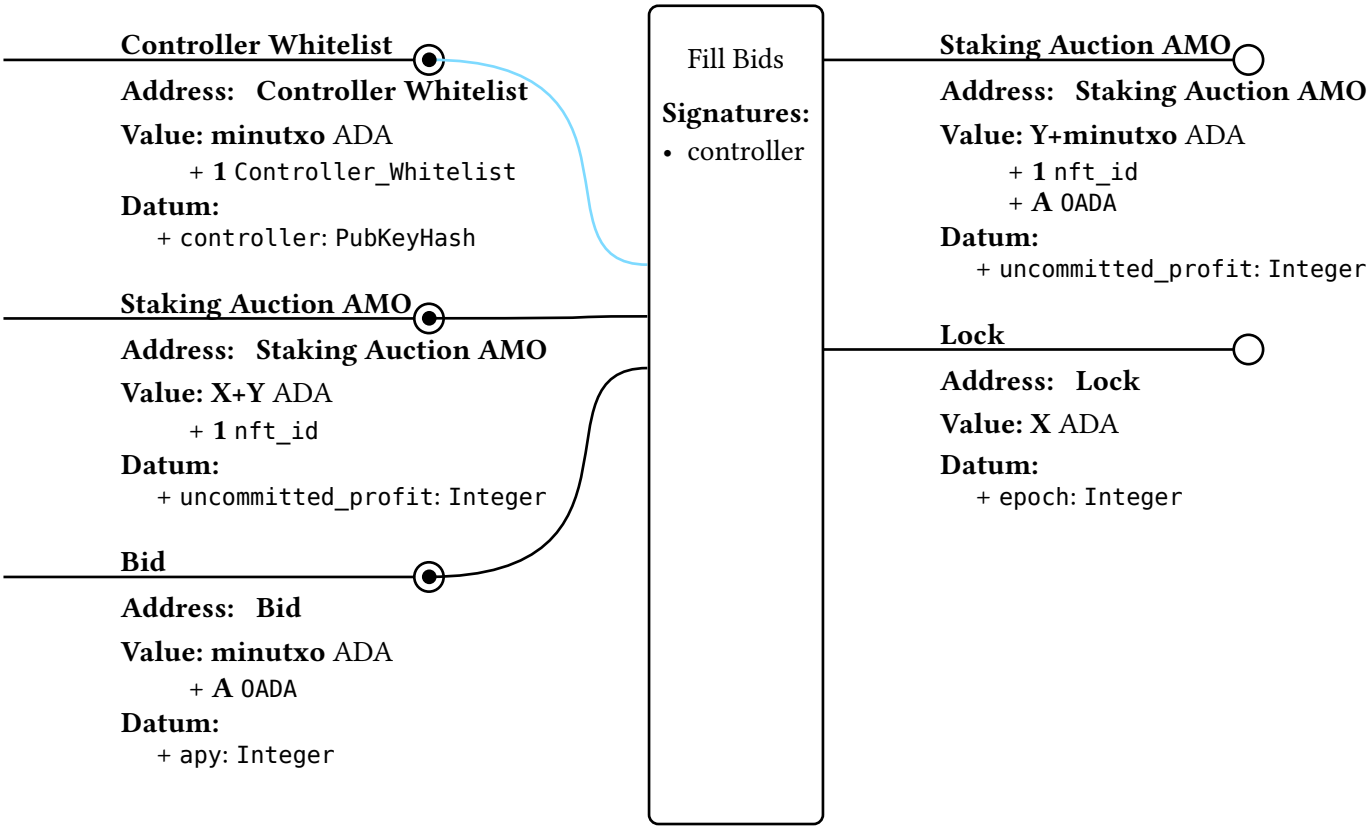


8.28. Partial Fill a Partial Bid



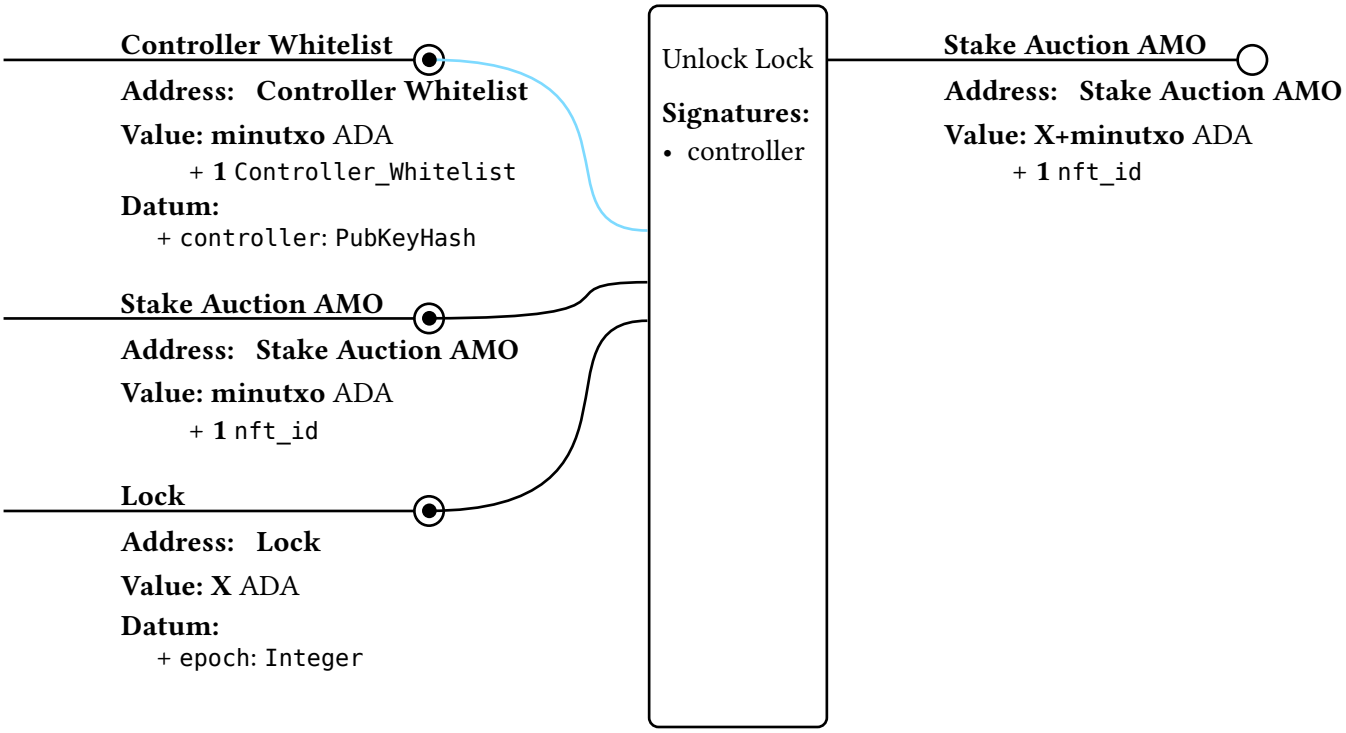
Note:
Can partial/fill multiple bids at once
X = B / (apy/73)
Lock has the current epoch in datum
uncommitted_profit += A

8.29. Fill a Fill or Kill Bid



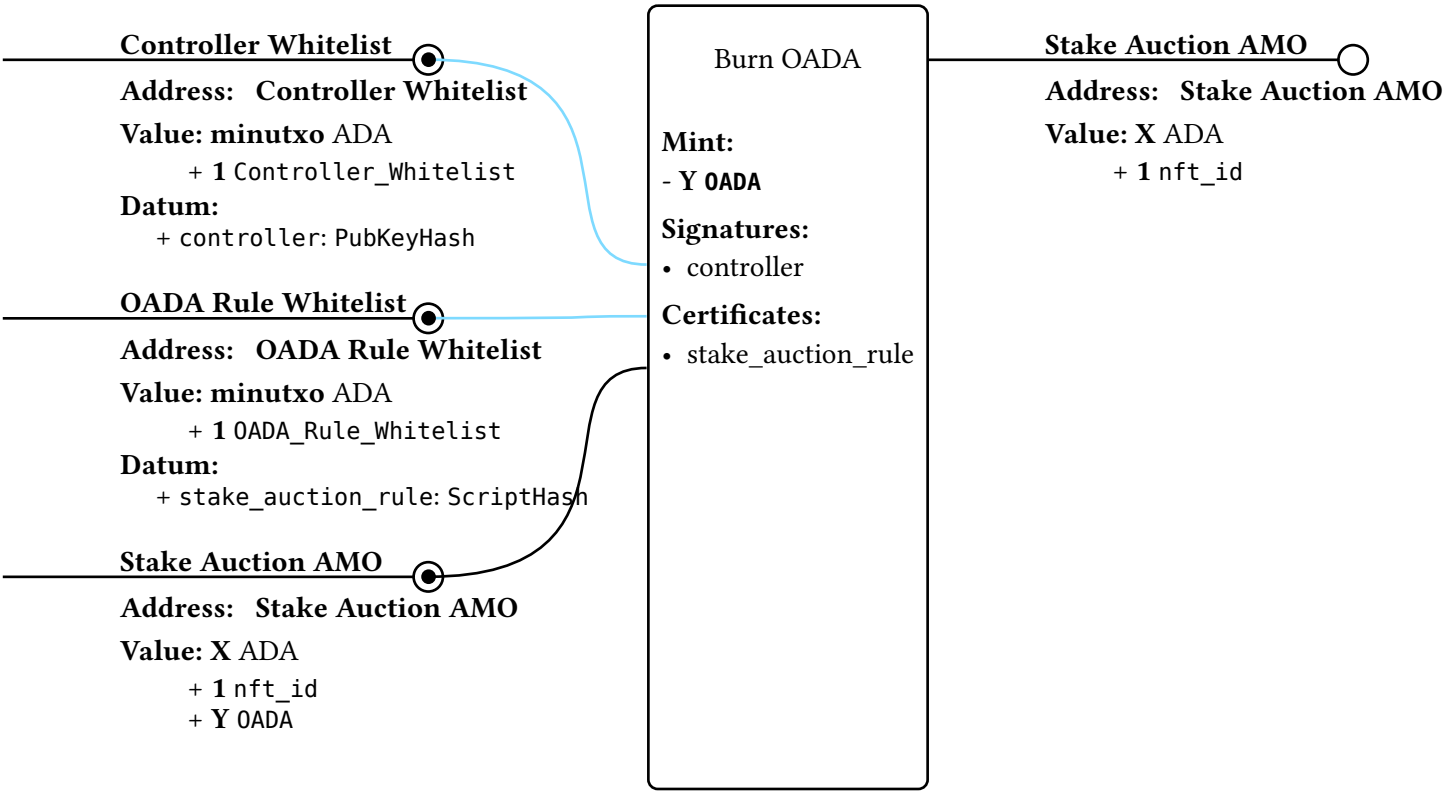
Note:
Can partial/fill multiple bids at once
X = B / (apy/73)
Lock has the current epoch in datum
uncommitted_profit += A

8.30. Process expired Stake Locks



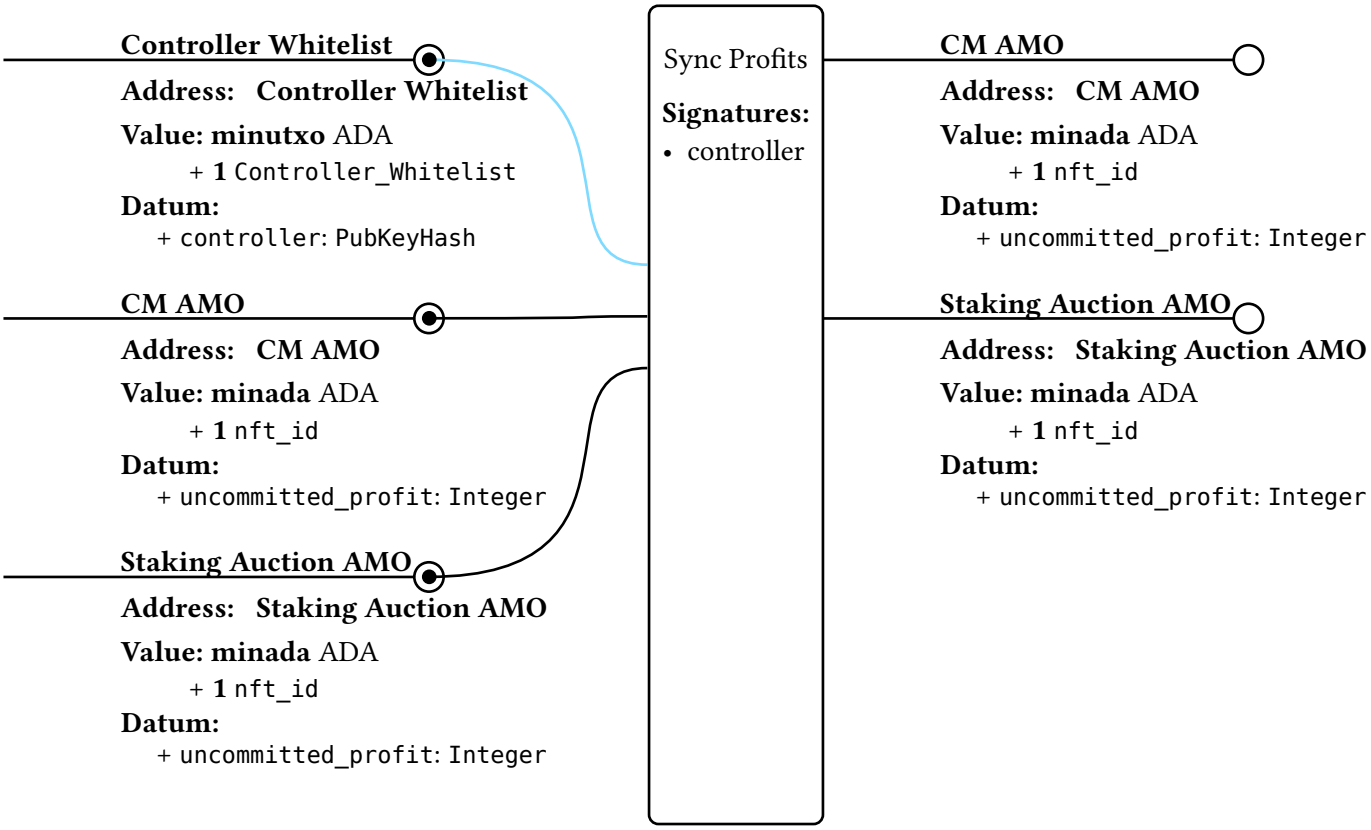
Note: Can unlock multiple at a time

8.31. Burn OADA from Bids

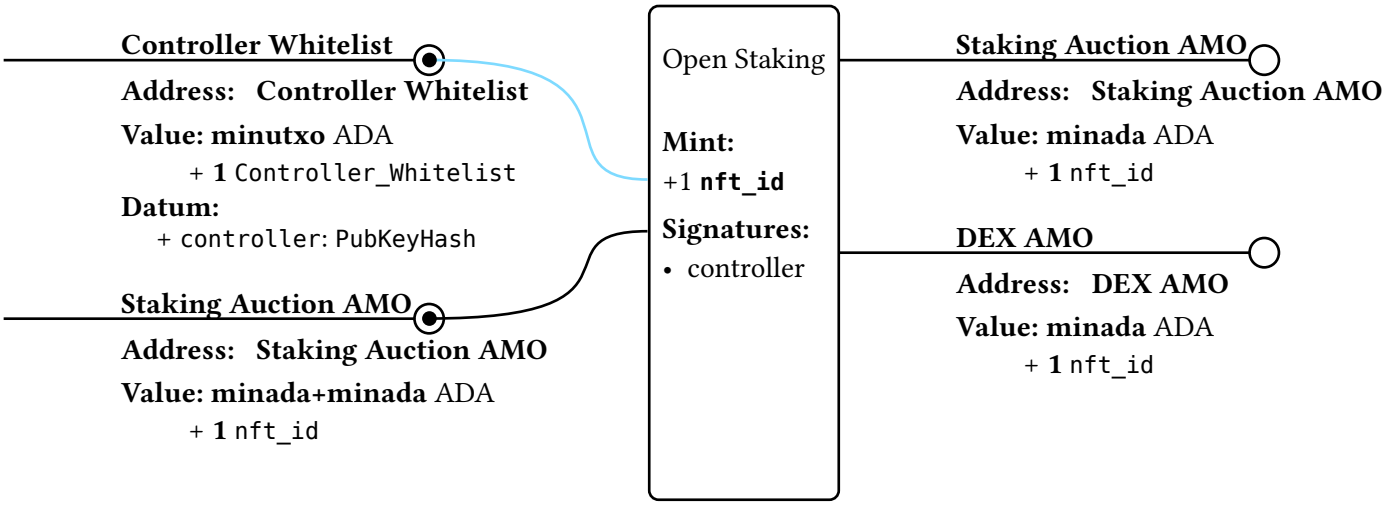


Note: Can unlock multiple at a time

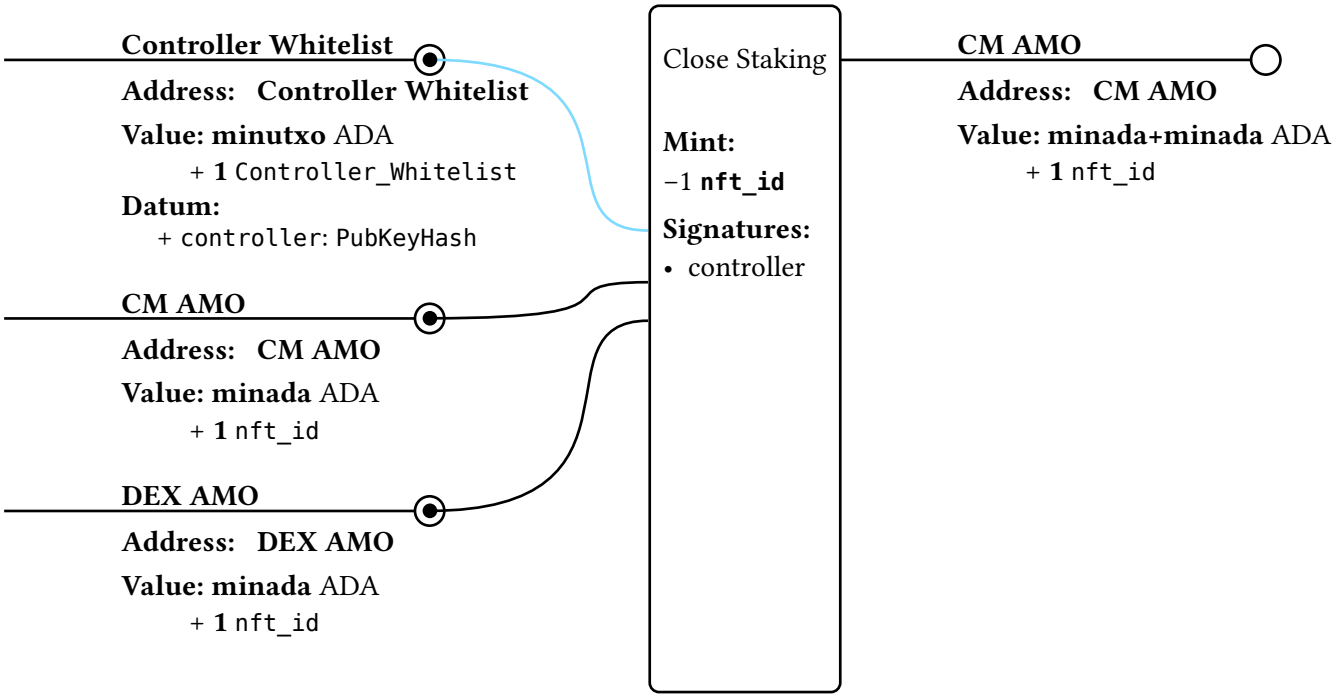
8.32. Sync Staking AMO Profits with the CM AMO



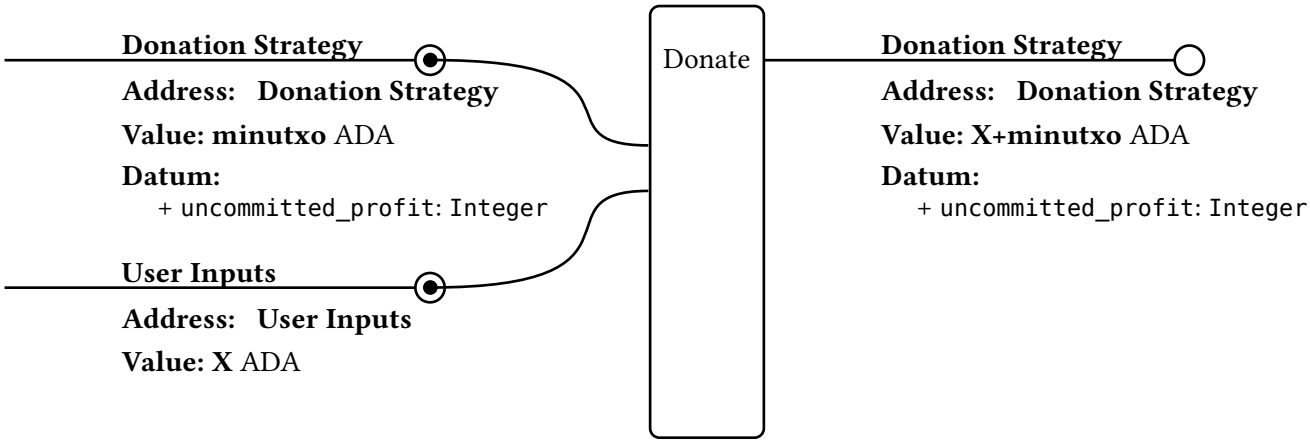
8.33. Open a Staking Auction AMO



8.34. Close a Staking Auction AMO

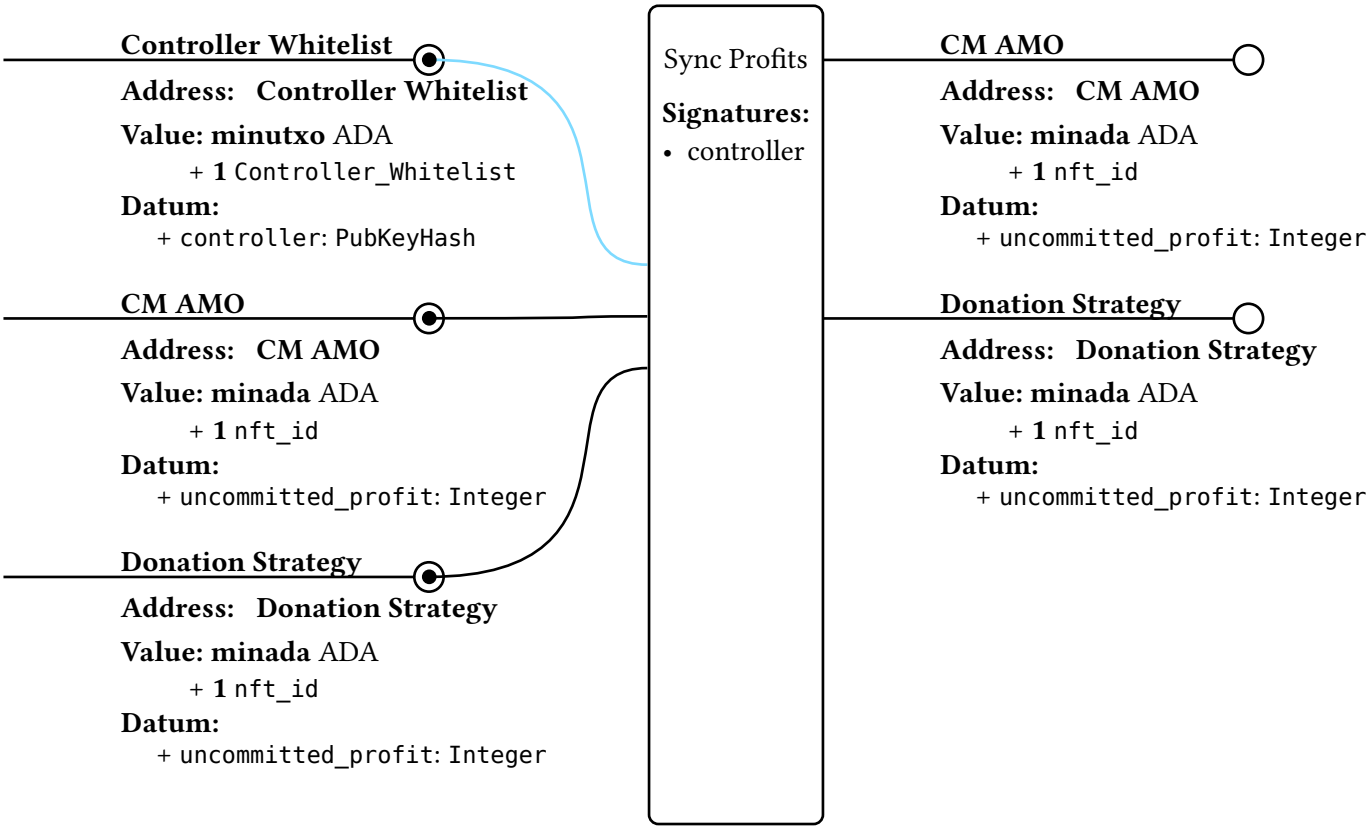


8.35. Donate via Donation Strategy

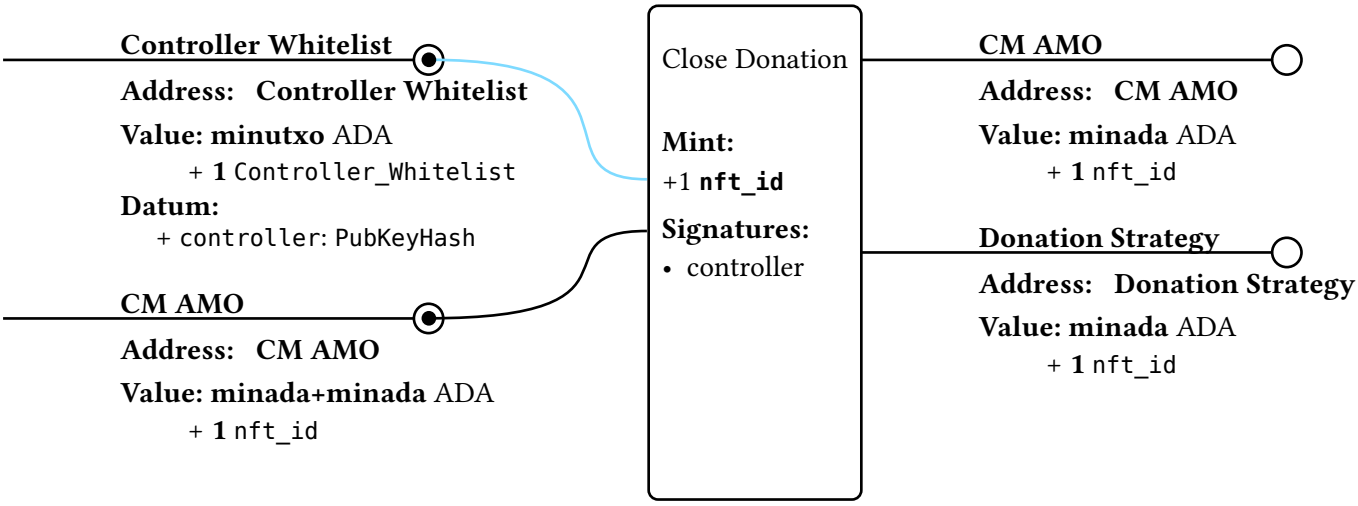


Note: uncommitted_profit += X

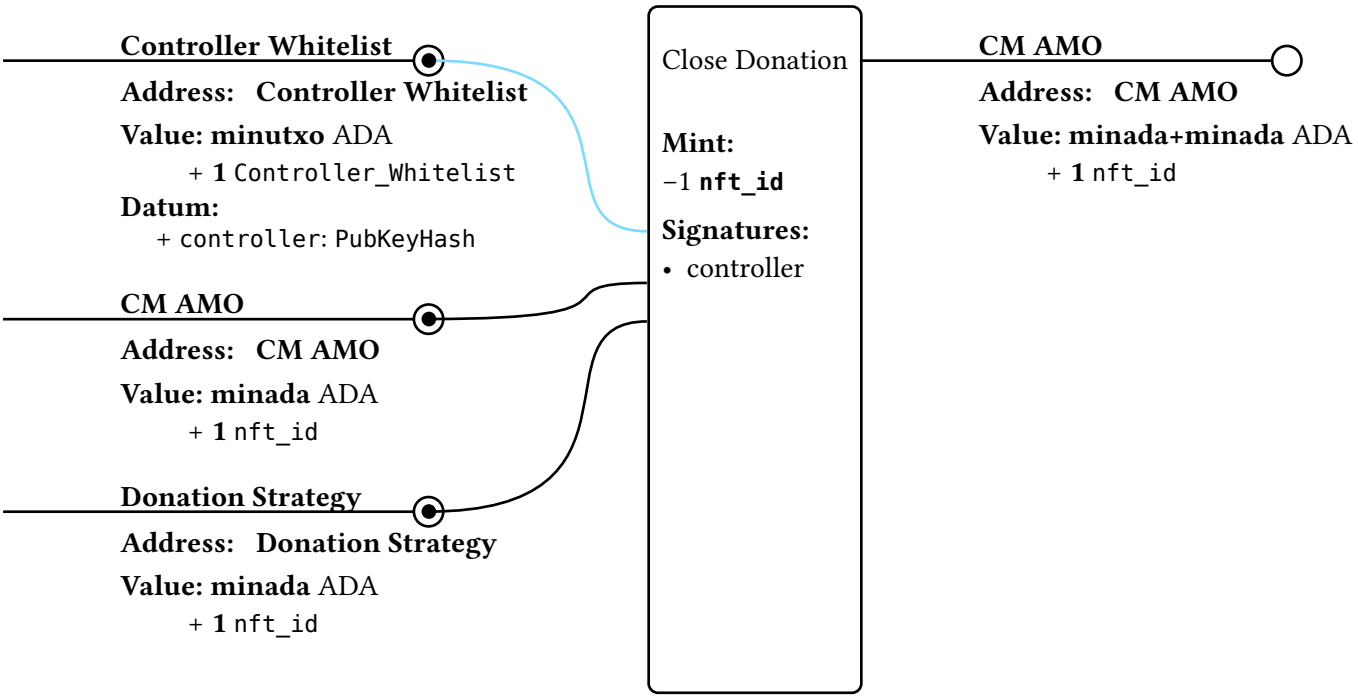
8.36. Sync Donation Profits



8.37. Open a Donation Strategy



8.38. Close a Donation Strategy



9. Glossary of Terms

9.1. OADA
A fully backed, 1:1 AD -pegged derivative token minted by depositing ADA. Acts as the core asset of the OADA system and is used across various staking, liquidity, and governance modules.

9.2. sOADA
The staked version of OADA, representing a user's share of system profits. Its exchange rate relative to OADA reflects cumulative yield earned through system strategies.

9.3. AMO (Algorithmic Market Operation)
A rule governed strategy module authorized to mint/burn OADA or deploy system capital into yield generating activities. Examples include DEX AMO, Staking Auction AMO, and Collateral Management AMO.

9.4. Controller
A governance approved agent (via whitelist) responsible for triggering capital movements, yield deployments, profit syncs, and peg keeping operations across the OADA system.

9.5. ODAO (Optim DAO)
The governing decentralized organization that oversees system rules, parameters, upgrades, and controller permissions. It is managed via a soul bound governance token.

9.6. Collateral Management AMO
The central accounting and orchestration layer. Tracks capital inflows, profit realization, and ensures strategies conform to risk and solvency constraints.

9.7. Staking AMO
Manages the minting and burning of sOADA. Calculates and updates exchange rates and facilitates profit distribution to stakers.

9.8. Stake Auction AMO
A continuous Dutch auction mechanism that sells staking rights to the highest bidders each epoch, maximizing yield while preserving principal solvency.

9.9. DEX AMO
A liquidity and peg management strategy that interacts with the Splash stableswap pool to stabilize OADA/ADA price via buybacks, sells, and LP provisioning.

9.10. Donation Module (UNHCR)
An extension of the OADA system enabling users to donate yield (not principal) to humanitarian causes like UNHCR. Operates within existing AMO frameworks and offers NFT based proof of impact.

9.11. NFT Impact Certificate
A non fungible token issued on unstake, recording the amount of donated yield and serving as a verifiable, on-chain acknowledgment of a user's philanthropic contribution.

9.12. Soul Token
A special governance token that enables protocol upgrades, AMO rule additions, and high level administrative actions. It is required for modifying core system behavior.

9.13. Whitelist

A registry of authorized actors (controllers, strategies, rules) that the system uses to gate permissioned operations such as minting, staking, or yield deployment.

9.14. Yield Donation Rate

A user selected percentage (25%, 50%, 75%, or 100%) of generated yield that is routed to UNHCR from the user's deposited capital.