

2023-02-22

Butane

Pro-rata Sale

π Lanningham



Sundae Labs

Contents

1 - Audit Manifest	3
2 - Context	4
3 - Specification	5
3.a - Detailed Specification	7
4 - Findings Summary	18
BTN-000 - No commitment to price or allocation by Admin	19
BTN-001 - Admin can change terms during partial claim or deadlock unclaimed funds	20
BTN-100 - Not commitment to sale schedule	21
BTN-101 - Incomplete solutions to previous findings	22
BTN-200 - Funds could get deadlocked if the recipient is a script	23
BTN-201 - Depositors must sacrifice their staking rewards	24
BTN-202 - Will not use state machine	26
BTN-300 - Attach staking addresses to deposits	27
BTN-301 - Issues with minUTXO protections	28
5 - Appendix	30
5.a - Disclaimer	30
5.b - Issue Guide	31
5.c - Revisions	33
5.d - About Us	33

1 - Audit Manifest

Please find below the list of pinned software dependencies and files that were covered by the audit.

Software	Version	Commit
Pro-rata Sale	0.1.0	c9fd48a2cc6ef6321c8d43face1edfe5bdd9f83f

Filename	Hash (SHA256)
validators/sale.ak	180694fc6b97b4b89474d44e838bc4a49f8587d3b55af9e5684a6ddf7ae8fc9c
validators/nft.ak	e50a6642c85499da9ccfa18204b4c63d9455a7edde86726746017eef98a4acbf
lib/launchpad/prorata.ak	b141707194119480c4ab4df48acfc941534e6b52d6ba1718a57834ddef0fc37e
lib/launchpad/types.ak	9c322668b8e8c034ede0ecaca95df242c5de66b9b0bef1046fc6bfd695c77496
lib/launchpad/utils.ak	512c6c801f07d6b8ae5d200fc48326b45b36f1161f7e25b278401cc695e6b8ae

Parameter	Value
init_oref	9fe7265c1022c6a68af14e7d4063b0f7a7832c58255a4a8d219f6fc5285c310a#0
target_recipient	addr1vye93uefq8r6ezk0kzq443u9zht7ylu5nelvw8eracd200q8t5935
admin_pol	26cd051dbdad984cda7e624a25aecc1bab0c97e206b3fb98a97b4b8
admin_tn	SaleAdmin

Validator	Method	Hash (Blake2b-224)
sale	deposit	f2f2041123c6a5bbb2eff268291512bce5fba3424ef63988b330313d
sale	collect	f2f2041123c6a5bbb2eff268291512bce5fba3424ef63988b330313d
nft	nft	26cd051dbdad984cda7e624a25aecc1bab0c97e206b3fb98a97b4b8

2 - Context

The Butane token is a new cardano native token with utility in the upcoming “Butane” synthetics protocol. The scripts under audit are a set of smart contracts that govern a so-called “pro-rata” sale of these tokens. The butane team intends to offer 43% of the total supply (10,750,000 Butane) to the public in this sale, at a rate of 0.9 ADA per Butane.

If less than 9,675,000 ADA is raised, the remaining tokens will be burned, reducing the total supply. If more than 9,675,000 ADA is raised, then each user will use the same percentage of their deposit to buy the token at that rate, and the remaining ADA will be returned to them.

For example, if 10,000,000 ADA is raised, then only 96.75% of the raised ADA can actually purchase tokens. Thus, a user who deposited 1,000 ADA would use 967.5 ADA to purchase 1075 Butane, and 32.5 ADA would be returned.

This audit was performed by Sundae Labs, with the following understanding:

- There would be an “admin” user that should be acknowledged as a point of centralization, and should be treated as a largely trusted actor.
 - That is, this actor should never have direct access to the user funds, but can be trusted to progress the protocol, and provide accurate off-chain global state.
- Performance of the contracts is not a factor.
- To limit scope and complexity, we are only auditing the Sale portion, not the Butane minting policy.
- The duration in which the sale should be active is limited, minimizing the window of time in which an attacker has to find a vulnerability.
- The contracts will remain closed source during the sale, minimizing the information available to an attacker during that vulnerable window.

3 - Specification

The butane prorata sale protocol is intended to facilitate the initial public sale of the Butane token.

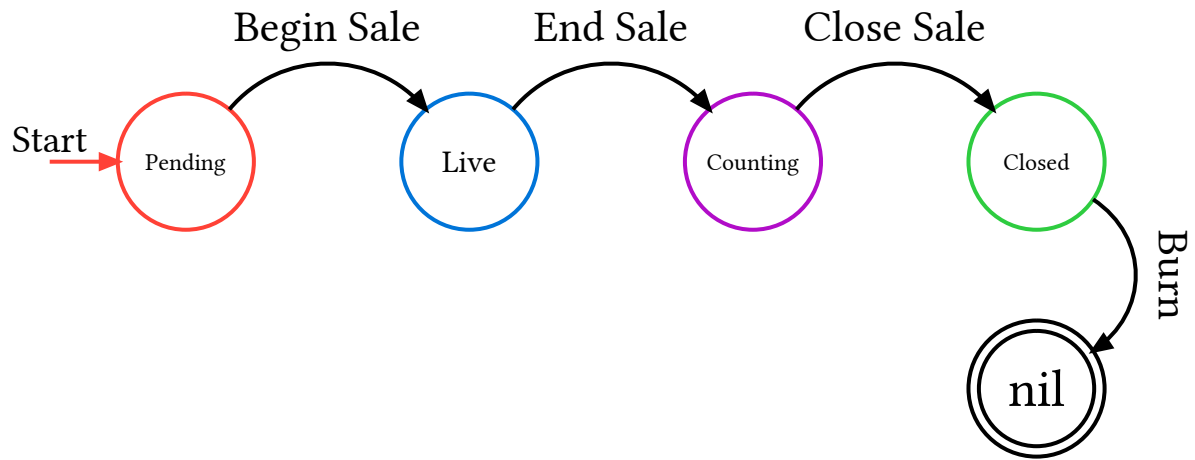
The high level objectives of this sale are:

- A trusted admin begins the sale
- While the sale is active, users can deposit funds in a bid to purchase Butane tokens
- This deposit must be a minimum amount of ADA
- At the end of the sale, the admin can close the sale, preventing further deposits
- The admin can then calculate the total subscription of the sale, and lock in a “pro-rata” percentage
- Users, or the admin in large batches, can then claim their portion of the sold tokens, paying a fixed rate for those they mint, and receiving the remaining amount of ADA in return.
- When rounding, values should be rounded in favor of the protocol. That is, a user may receive up to 1 diminutive unit less of Butane than they paid for, resulting in a slightly smaller total supply of Butane.
- The admin can exercise discretion over the timing of the raise
- The admin should not be able to change the terms of the sale
- Users should be able to reclaim their ADA after an expiration if the sale doesn’t terminate for some reason.

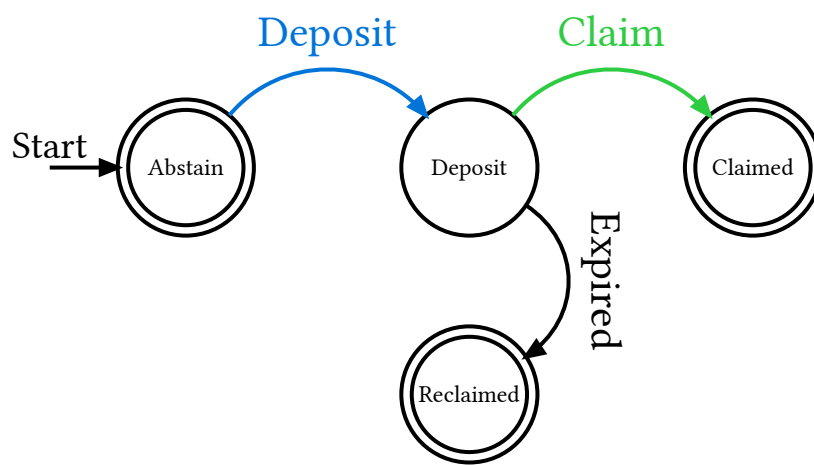
In particular, what we mean by pro-rata distribution is:

- Let the butane price be the quantity of butane per 1 ADA raised
- Let the total purchased butane amount be the total deposited ADA multiplied by the butane price
- If the total purchased butane amount is less than the amount allocated to the sale
 - Let each deposit be spent if the deposited ADA is paid to the admin, and $\text{deposit_amount} * \text{butane_price}$ is minted and paid to the depositer
- If the total purchased butane amount is greater than the amount allocated to the sale
 - Let the pro-rata percentage be $\text{sale_allocation} / \text{total_purchased_butane_amount}$
 - Let the pro-rata payment be $\text{deposited_ada} * \text{pro_rata_percentage}$
 - Let the rebate amount be $\text{deposited_ada} - \text{pro_rata_payment}$
 - Let each deposit be spent if pro_rata_payment ADA is paid to the admin, and $\text{pro_rata_payment} * \text{butane_price}$ Butane plus rebate_amount ADA is paid to the depositer

The full protocol should adhere to the following state transition diagram:



Each depositing user should adhere to the following state transition diagram. The highlighted transitions are only valid when the sale is in the appropriately colored state above.



3.a - Detailed Specification

3.a.i - Definitions

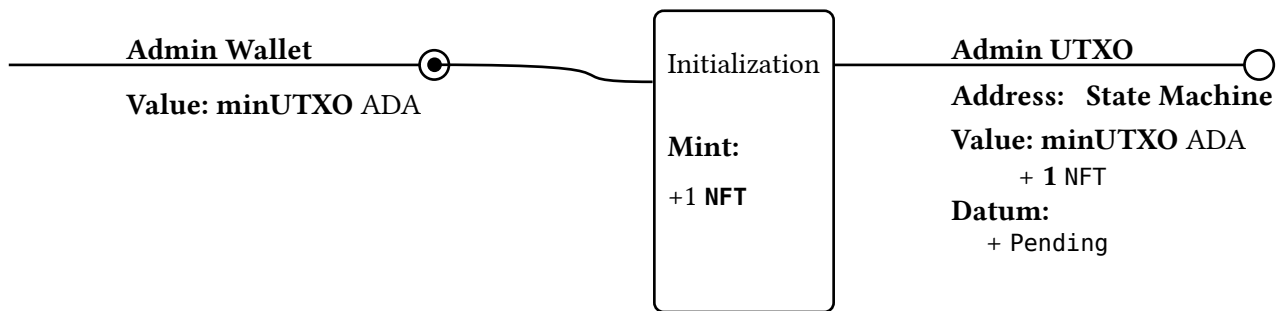
- **Butane Token**
 - The native token of the butane protocol being sold as part of these contracts
- **Public Sale**
 - A specific offering of the **Butane Token** available to all users through the decentralized protocol covered by this audit
- **Admin UTXO**
 - A trusted UTXO, authenticated by a unique NFT, used as a reference input to describe the global state of the **Public Sale**
- **Admin NFT**
 - The NFT authenticating the **Admin UTXO**
- **State Machine**
 - A smart contract implementing a simple state machine through which the **Admin UTXO** is advanced by the **Admin** .
- **Admin**
 - A user trusted to advance the **Admin UTXO** through the stages of the sale correctly.
- **Deposit**
 - Either:
 - The act of depositing ADA to participate in the sale
 - A specific UTXO holding the ADA of a user participating in the sale
- **Counting Token**
 - A temporary token minted to authenticate participation in the **Public Sale** .
- **Sale Allocation**
 - The total amount of **Butane Token** allocated to the **Public Sale**
- **Sale Price**

- The price of the **Butane Token** for the purposes of the **Public Sale**, either in ADA per Butane or Butane per ADA, depending on context.
- **Target Raise**
 - The total amount of ADA being sought as part of the **Public Sale**. Calculated as the **Sale Allocation** times the **Sale Price** in ADA per Butane.
- **Subscription**
 - The total amount of ADA deposited in the **Public Sale**
Related:
 - A **Public Sale** is under-subscribed when the total amount of ADA raised is less than the **Target Raise**
 - A **Public Sale** is over-subscribed when the total amount of ADA raised is greater than the **Target Raise**
- **Pro-rata Percentage**
 - The percentage of the **Sale Allocation** that each user is entitled to, based on their deposit amount. This is calculated as follows:
 - If under-subscribed, it is 100%
 - If over-subscribed, it is the **Sale Allocation** divided by the **Subscription**.
- **Rebate**
 - Some amount of ADA returned returned to the user when the **Public Sale** is over-subscribed.
- **Target Recipient**
 - The cardano address that will receive ADA raised from the **Public Sale**
- **Recipient**
 - The address attached to a specific deposit which will receive the **Butane Token** and possibly a **Rebate**.

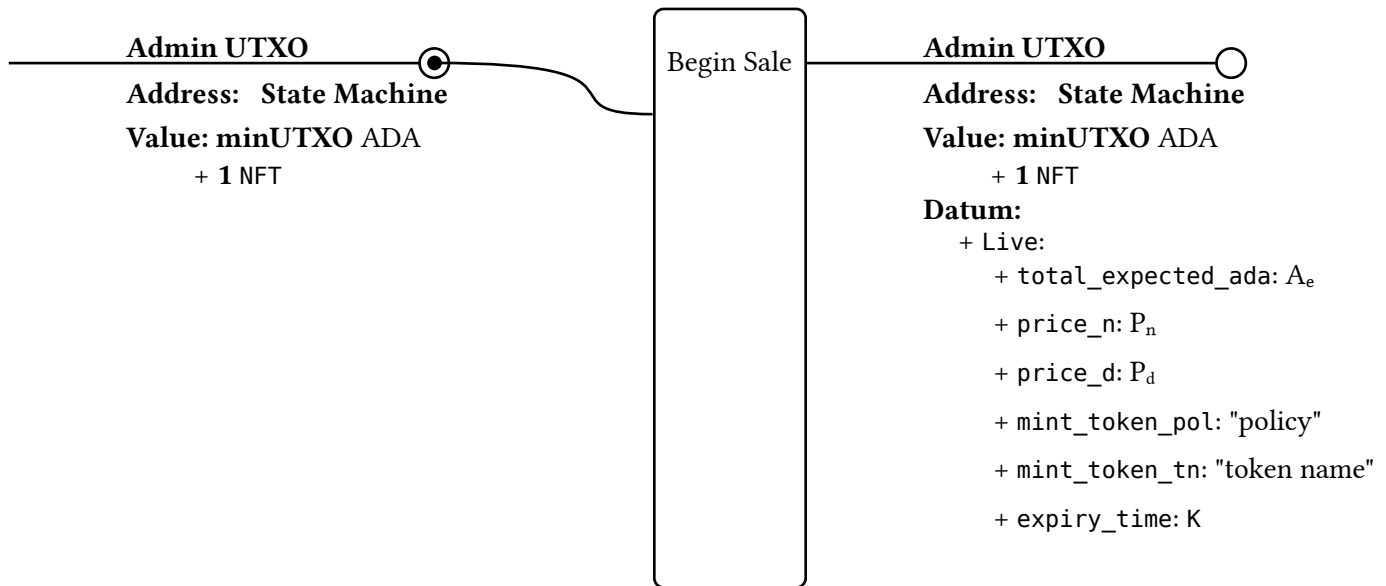
3.a.ii - Transactions

There are 7 transaction archetypes in this protocol. We present each in detail below.

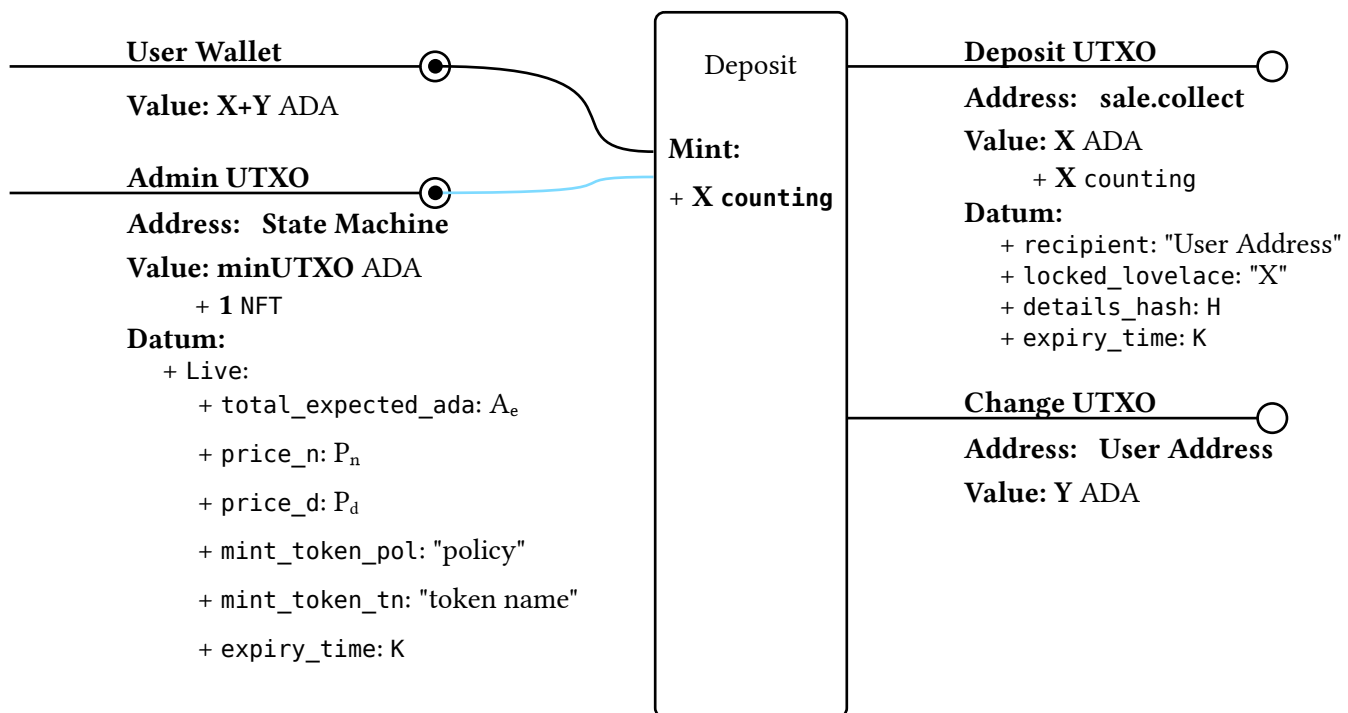
- Initialization
 - A new **Public Sale** is initialized by the **Admin** creating the **Admin UTXO**
 - An NFT must be minted and paid into a UTXO with the correct Pending datum.
 - This represents the initial Start transition in the state diagram above.



- Begin Sale
 - The Admin opens the sale to the public by spending the Admin UTXO and updating the state to Live.
 - Represents the transition from “Pending” to “Live” in the diagram above.
 - Involves a commitment to sale details.

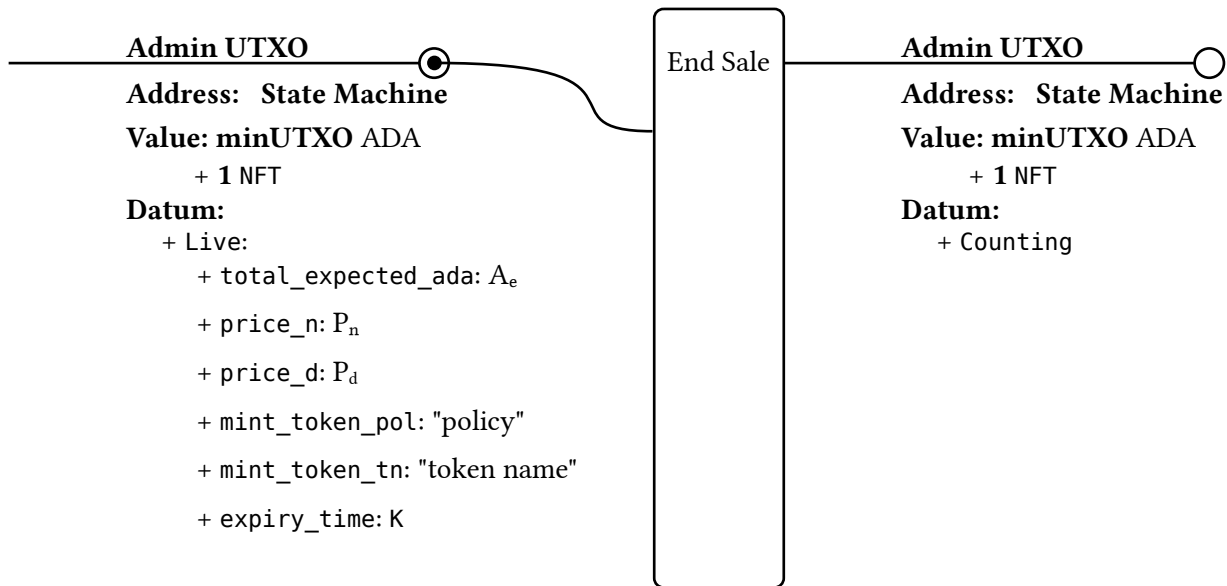


- Deposit
 - A user deposits ADA, securing their position in the **Public Sale** .
 - An equal amount of **Counting Token** is minted to verify that the deposit was made while the **Public Sale** was still ongoing, rather than after the fact.
 - The **Public Sale** must be live, as proven through a reference input.
 - The user must commit to the details of the sale.
 - Represents the users transition from “Abstain” to “Deposit” in the diagram above.

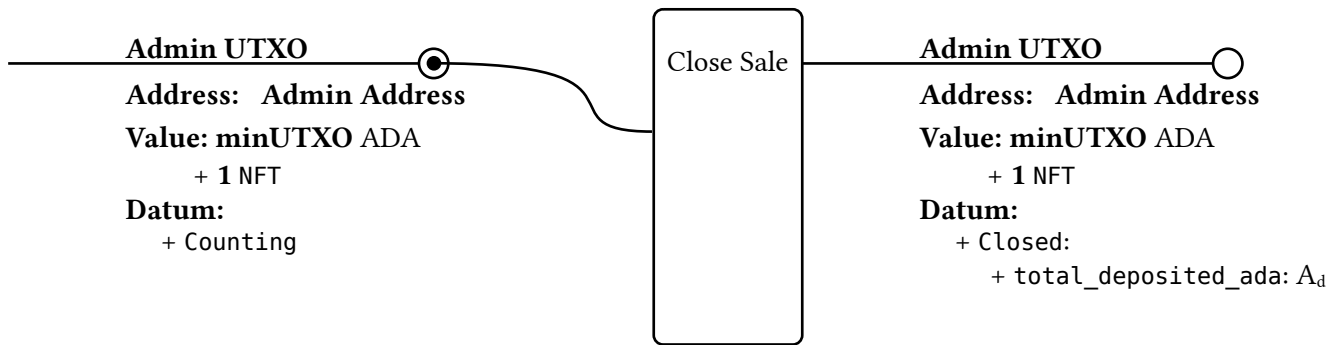


Note: H is the hash of the admin Datum launch details.

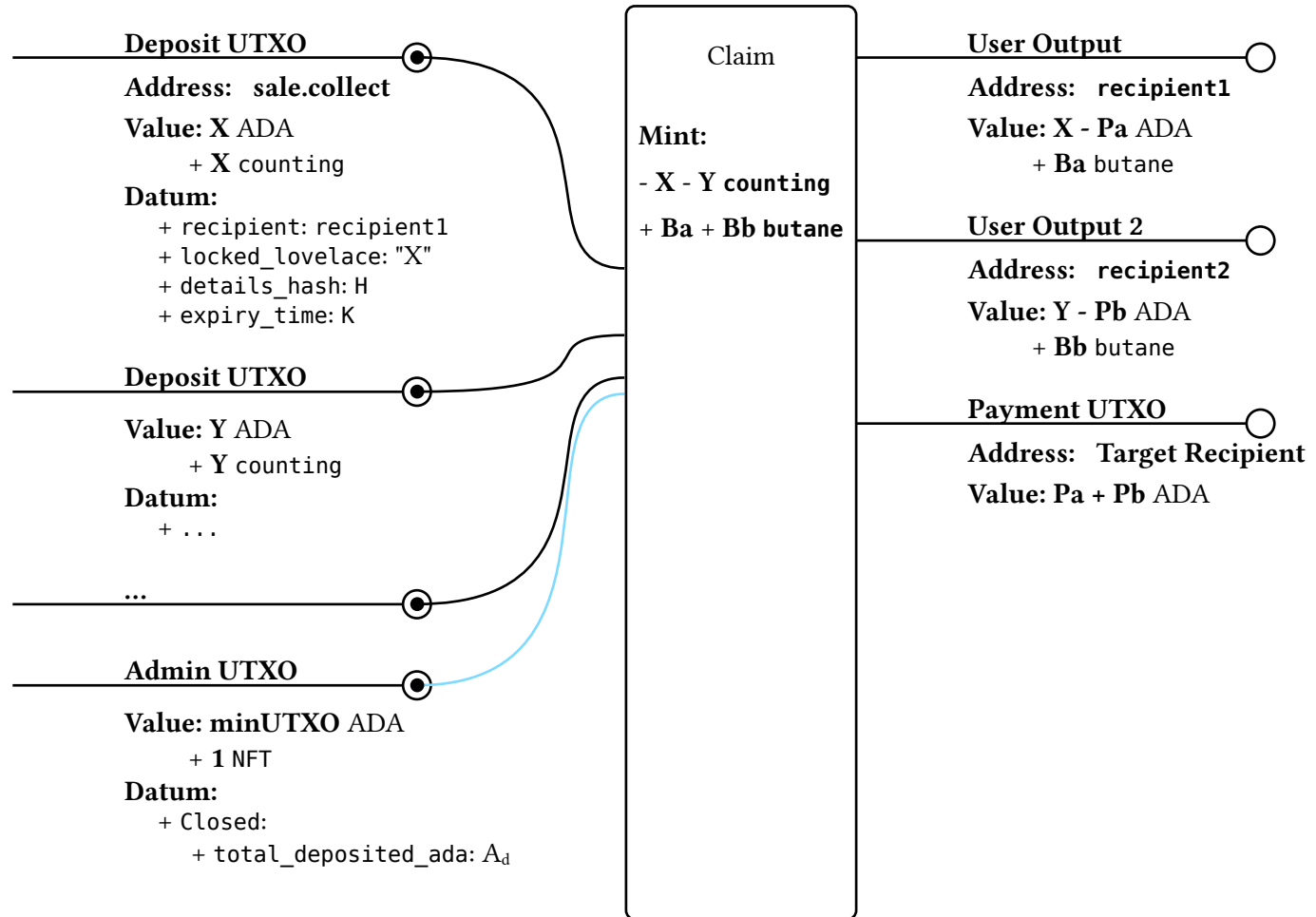
- End Sale
 - End the **Public Sale**, preventing further deposits
 - Also allows time for the **Admin** to calculate the **Subscription** and **Pro-rata Percentage** before closing the sale.
 - Represents the transition from “Live” to “Counting” in the diagram above.



- Close Sale
 - The Admin closes the sale after calculating the Subscription and Pro-rata Percentage .
 - Locks in a price and a Subscription amount, so that claims can begin claiming.
 - Represents the transition from “Pending” to “Closed” in the diagram above.



- Claim
 - Someone (either the user, or someone on their behalf) spends a deposit, burning the **Counting Token**, paying the appropriate ADA to the **Target Recipient**, and minting the appropriate amount of **Butane Token**, paid along with any **Rebate** to the **Recipient**.
 - $\frac{P_n}{P_d}$ is the **Sale Price** denominated in ADA per Butane
 - A_e refers to the **Sale Allocation**
 - A_d refers to the **Subscription**



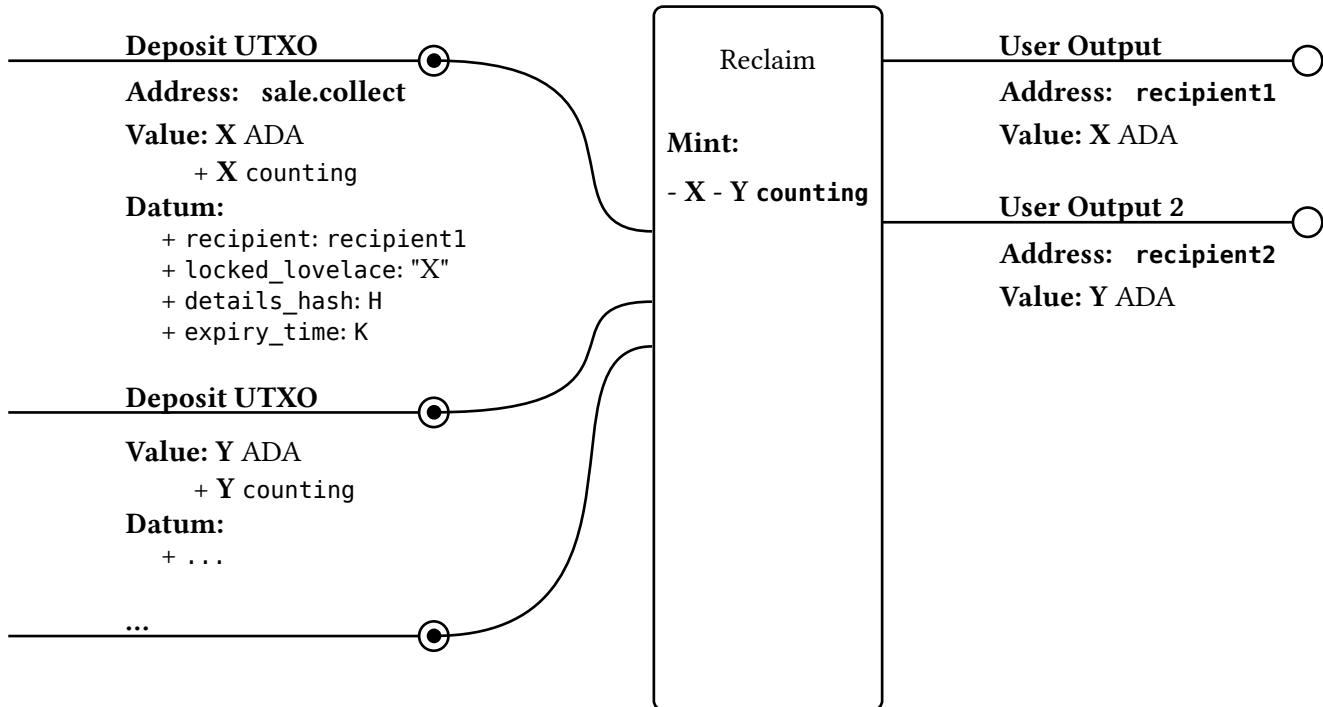
Note:

H must match hash of details in each deposits redeemer

$$P_i = \begin{cases} X_i & \text{if } A_d \leq A_e \\ \left\lfloor \frac{X_i * A_e}{A_d} \right\rfloor & \text{if } A_d > A_e \end{cases}$$

$$B_i = \left\lfloor \frac{P_i * P_d}{P_n} \right\rfloor$$

- Reclaim
 - Someone reclaims their ADA after the expiration.
 - Only used in disaster recovery, if the sale datum is burned too soon.



Note:

K must be before the transaction lower bound.

3.a.iii - Core Invariants

We use the following labels for the terms defined above:

- $\frac{P_n}{P_d}$ is the **Sale Price** denominated in ADA per Butane
- A_e refers to the **Sale Allocation**
- A_d refers to the **Subscription**

The protocol is considered correctly executed if:

- 1) The **Admin NFT** is minted and paid with a Pending datum to the state machine script.
- 2) The **State Machine** is advanced to Live with sale details matching those published publicly.
- 3) The **State Machine** is advanced by the **Admin** according to a reasonable schedule.
- 4) At least 1 hour without rollbacks passes between advancing the **State Machine** to Counting, and advancing it to Closed.
- 5) The **Subscription** amount provided to the state machine represents the total supply of the **Counting Token**, and thereby the total amount of ADA locked in the script.

The following core invariants should hold:

- 1) The **Admin** can be trusted to progress the protocol, and provide accurate off-chain global state.
- 2) A user can only mint an amount **Counting Token** by paying the same amount of Lovelace into the `sale.collect` script
- 3) A user must lock at least 300 ADA to participate in the sale.
- 4) A user can only mint Butane by burning the full amount of **Counting Token** on their UTXO
- 5) A user can only mint **Counting Token** while the **Admin UTXO** is Live
- 6) A user can only claim the results of the **Public Sale** after the **Admin UTXO** has progressed to the Closed state.
- 7) The **Admin** cannot change the details of the sale after it is in the Live state.
- 8) The total minted butane minted must not exceed $A_e * \frac{P_d}{P_n}$
- 9) A user can only mint Butane by paying $\frac{P_n}{P_d}$ ADA per **Butane Token** paid to the **Target Recipient**
- 10) The **Butane Token** minted from a deposit must be paid to the **Deposit Recipient**
- 11) If $A_d \leq A_e$, a user must pay the full amount of `locked_lovelace` to the **Target Recipient** and mint $\left\lfloor \frac{\text{locked_lovelace} * P_d}{P_n} \right\rfloor$ Butane to the **Recipient**
 - a) We use floor because we cannot mint a fractional diminutive unit, and rounding down favors the protocol: the user may receive 1 diminutive unit less Butane than they paid for
 - b) This helps maintain Invariant 5
- 12) If $A_d > A_e$, a user must pay `paid_lovelace` = $\left\lfloor \frac{\text{locked_lovelace} * A_e}{A_d} \right\rfloor$ ADA to the **Target Recipient** and mint $\left\lfloor \frac{\text{paid_lovelace} * P_d}{P_n} \right\rfloor$ Butane to the **Recipient**
 - a) We take the floor when calculating the `paid_lovelace` to maintain Invariant 5. If we took the ceiling, `paid_lovelace` might be 1 lovelace more than the correct pro-rata amount, which could result in higher minted Butane than the **Sale Allocation**
 - b) Similarly, we take the floor when calculating the minted Butane to maintain Invariant 5

- 13) Any difference between `locked_lovelace` and `paid_lovelace` must be returned to the **Recipient** along with the minted Butane.
- 14) After some minimum time delay, if the **Deposit** hasn't been claimed yet, a user can reclaim their `locked_lovelace` from the `sale.collect` script

4 - Findings Summary

ID	Title	Severity	Status
<u>BTN-000</u>	No commitment to price or allocation by Admin	Critical	Resolved
<u>BTN-001</u>	Admin can change terms during partial claim or deadlock unclaimed funds	Critical	Resolved
<u>BTN-100</u>	Not commitment to sale schedule	Major	Acknowledged
<u>BTN-101</u>	Incomplete solutions to previous findings	Major	Resolved
<u>BTN-200</u>	Funds could get deadlocked if the recipient is a script	Minor	Resolved
<u>BTN-201</u>	Depositors must sacrifice their staking rewards	Minor	Acknowledged
<u>BTN-202</u>	Will not use state machine	Minor	Identified
<u>BTN-300</u>	Attach staking addresses to deposits	Info	Acknowledged
<u>BTN-301</u>	Issues with minUTXO protections	Info	Acknowledged

BTN-000 - No commitment to price or allocation by Admin

Category	Commit	Severity	Status
None	53cb47dc77da6a9c053d2ab9c8baf416ea1e7409	Critical	Resolved

Description

The protocol has an Admin role, that has the following two trust assumptions:

- They will progress the protocol through the state machine
- In the Close sale transaction, they will provide an accurate value for A_d , such that A_d is the sum of all locked_lovelace in valid **Deposit** s.

However, the admin has much more control and requires much more trust than this. In particular, there is no commitment to the total expected ADA, the sale price, or the butane token policy ID.

This makes the following attacks possible, which we deem outside of a reasonable trust assumption for such a sale:

- Users lock up ADA in expectation of receiving Butane at a price of 0.9 ADA per Butane, and instead the price is set to 1000 ADA per Butane.
- Users lock up ADA in expectation of collectively receiving 43% of the supply, and instead only 0.1% of the supply is distributed, with the rest retained by the team.
- Users lock up ADA in expectation of receiving Butane, but are given some other worthless token instead.

Recommendation

Add `total_expected_ada`, `price_n`, `price_d`, `mint_token_pol` and `mint_token_tn` to the Pending and Live datums.

Lock the Admin UTXO in a script that allows the **Admin** to adhere to the state changes, and enforces that these values remain the same during each state transition.

This way, users can confirm the details of the sale before making the decision about whether to participate.

Resolution

This issue was resolved as of commit 53cb47dc77da6a9c053d2ab9c8baf416ea1e7409.

BTN-001 - Admin can change terms during partial claim or deadlock un-claimed funds

Category	Commit	Severity	Status
None	bcddbd7a2a590404db8e6abbf0599b20b9b608eb	Critical	Resolved

Description

The admin is trusted to progress the protocol, and provide accurate information.

However, in the current implementation, because the Admin UTXO is just held in a wallet, after the sale is closed, the admin can spend the UTXO before all participants have claimed.

First, this could be used to change the terms of the sale part way through claims. For example, the admin could immediately claim funds for any insiders immediately when the sale is closed, and then change the price or allocation for the remaining claimants.

Second, the admin could spend the UTXO to remove the datum completely. Because claiming depends on this UTXO as a reference input, this would permanently lock users ADA in the deposit.

Recommendation

As with the other findings, lock the Admin UTXO in a script. The script should enforce that after the sale is closed and the terms are provided, it is never spent again.

This ensures that no user funds become deadlocked, and the terms of the sale don't change after some users have claimed.

Alternatively, allow users to reclaim their ADA after some timeout, to ensure that even if the Admin behaves badly, at the very least they get their ADA back.

Resolution

This issue was resolved as of commit bcddbd7a2a590404db8e6abbf0599b20b9b608eb.

BTN-100 - Not commitment to sale schedule

Category	Commit	Severity	Status
None		Major	Acknowledged

Description

The protocol assumes that the admin will be responsible for eventually progressing the protocol, but does not assume they are trusted to progress the protocol in accordance with a schedule.

One design goal is to allow reasonable flexibility in the timing. This is to allow an **almost** subscribed sale to wait for it's full sale before closing, without prescribing a set time.

However, in the current protocol, the admin must be trusted to adhere to that reasonable schedule. For example, the admin could drag out the sale for months if the sale is under-subscribed, trying to eke out more and more ADA, while participants are locked in and have no way to cancel.

Similarly, the admin could close the sale at any time, limiting the participants and impacting the amount of Butane burned.

Finally, the admin could reopen the sale after some tokens have been claimed, allowing further deposits to raise more ADA.

Recommendation

Simplify the sale state machine to just include Sale and Closed states. Lock the Admin UTXO in a script that allows the admin to progress the protocol. Add a `start_time`, `minimum_length` and `maximum_length` fields to the Sale and Closed datums. Use the `start_time` to prevent deposits before the sale starts, and `start_time + maximum_length` to automatically close the sale and prevent further deposits. Then, only a single transition from Sale to Closed is needed from the admin to report the Subscription. Enforce that this happens after `start_time + minimum_length` to ensure the sale stays open for a minimum length of time. Add the ability to reclaim ADA if they wish after some delay. This way, if the protocol never progresses for some reason, at the very least the ADA isn't locked forever. In this way, the Admin has flexibility over the timing of the sale, but users also know ahead of time the reasonable bounds on the schedule of the protocol. The trust placed in the Admin is minimized, as they are only trusted to progress the protocol and provide an accurate Subscription amount.

Resolution

This issue was acknowledged by the project team with the comment: Flexibility in the timing of the sale is a design goal, and combined with the state machine and expiration we implemented for other findings, we believe the timing issue is safely within trust assumptions we are comfortable with

BTN-101 - Incomplete solutions to previous findings

Category	Commit	Severity	Status
None	c0f71e9952487cd491888bfb13ad2e78b11a89b0	Major	Resolved

Description

The solutions to **BTN-000** and **BTN-001** provide some protection, reducing the severity to only “Major”, but are not sufficient in our view.

BTN-000 was resolved by adding sale details to the Live datum, storing the hash of the sale details in the deposit datum, and verifying the sale details provided through the redeemer match the hash.

BTN-001 was addressed by implementing a simple state machine that transitions from Pending, to Live, to Closed.

However, the implementation suffers from two issues:

- The state machine never no longer has a transition back to Pending, meaning deposits cannot be halted to count up the total subscription. A user may deposit in between the total being counted and the sale being closed, resulting in an excess of Butane being minted.
- The state machine can be burned, deleting the admin token. Any unclaimed ADA would be permanently locked at that point.

Recommendation

- Add an intermediate state to allow for counting the total subscription in between the close of the sale and the opening of claims.
- Either disallow the evaporation of the state machine, or add an expiration, after which a user that was never claimed can reclaim their ADA, even if the admin UTXO has been burned.

Resolution

This issue was resolved as of commit c0f71e9952487cd491888bfb13ad2e78b11a89b0.

BTN-200 - Funds could get deadlocked if the recipient is a script

Category	Commit	Severity	Status
None	580bba7c85603cd87f6b1e037d30915ca0cf543f	Minor	Resolved

Description

If users construct their own transaction to participate in the sale, and direct the funds to a script address, they may become permanently locked.

First, the ADA could never be reclaimed, because the contracts explicitly expect `VerificationKeyCredentials`. Second, we enforce that the output when minting the butane has `NoDatum`. A script address with `NoDatum` on the UTXO cannot be spent.

We don't expect users to build their own transactions given the short time window when this will be active, but if these scripts are reused, this could be more of a concern.

Recommendation

One approach is to split the `Destination` and the `Owner` into two separate fields. The destination is an address and a datum, and the owner is a condition authorized to reclaim the ADA after some time.

A simpler approach is just to enforce that the recipient is a `VerificationKeyCredential` during the initial deposit.

Resolution

This issue was resolved as of commit `580bba7c85603cd87f6b1e037d30915ca0cf543f`.

BTN-201 - Depositors must sacrifice their staking rewards

Category	Commit	Severity	Status
None		Minor	Acknowledged

Description

A last minute change was made to the smart contracts to simplify the off-chain code.

In order to make tracking deposits easier, and allow enterprise addresses to participate, the contracts were changed after the conclusion of the audit, with the following condition:

```
assert(  
    locked_stakecred == Inline(deposit_recipient_pay),  
    @"Recipient payment cred and locked stake cred should be the same.",  
),
```

In particular, the **stake credential** of the locked deposit must be the users **payment credential**.

This reintroduces concerns from [BTN-300](#) and escalates them to a minor severity, as loss of staking rewards (if the sale extends over an epoch boundary) is no longer something that **could** happen, it is something that **must** happen.

Now, the user is forced to attach their payment credential in the staking address of the deposit. The sale is being conducted the day before a new epoch, and so if there is any delay in executing the sale, then staking rewards will likely be lost.

At full subscription, we estimate this to be about 6000 ADA in lost rewards.

A sophisticated user expecting to make a large purchase could register their stake address and delegate to a pool, but the vast majority of users are unlikely to have the sophistication to do this.

Recommendation

In order to allow enterprise addresses to participate, the condition could be relaxed as follows:

```
assert(  
    or {  
        locked_stakecred == None,  
        locked_stakecred == Inline(deposit_recipient_stake),  
    }  
    @"Locked stake cred must be none, or the recipient stake cred",  
),
```

Offchain code could index positions by the attached stake credential, which is also slightly safer with multi-address wallets.

Resolution

This issue was acknowledged by the project team with the comment: We see this lost ADA as an acceptable trade-off for the simplicity of the off-chain code, and for avoiding further last minute changes.

BTN-202 - Will not use state machine

Category	Commit	Severity	Status
None		Minor	Identified

Description

Due to various constraints, the team has decided not to use the state machine implemented as part of previous feedback, and to hold the admin UTXO in a normal wallet.

Ultimately the team feels, and we agree, that the other protections put in place provide significant safety for the user:

- If the terms of the sale in the admin UTXO are changed, the team will not be able to collect the ADA of any user who agreed to the sale under the original terms
- If the Admin UTXO is destroyed, users will be able to reclaim their ADA after the expiration period, which is also committed to at the time of lockup.

While this change does represent some amount of additional risk, one of the base assumptions already places the admin in a trusted role, and this doesn't materially change that.

Recommendation

If the team receives pushback on this, we recommend finding an independent third party to orchestrate the admin UTXO, or placing the admin UTXO into a multisig wallet.

BTN-300 - Attach staking addresses to deposits

Category	Commit	Severity	Status
None		Info	Acknowledged

Description

Because the sale may span an epoch boundary, you should ensure the users stake address is attached to any deposits. This doesn't impact the correctness of the script, but ensures that users earn staking rewards for any ADA before the sale officially completes.

You likely intended to do this anyway, but it's worth highlighting explicitly.

Resolution

This issue was acknowledged by the project team with the comment: We address this by enforcing the recipient stake address equals the deposit utxo stake address

BTN-301 - Issues with minUTXO protections

Category	Commit	Severity	Status
None		Info	Acknowledged

Description

One stated goal was to allow the user to lock up an arbitrary amount of ADA, so long as the purchase amount in the datum is less than or equal to the amount in the datum.

This is to allow the user to include the minUTXO amount, so that the Butane team doesn't need to cover the minUTXO amount when forcing claims for users.

However, there are three issues.

First, the user is not actually allowed to lock up an arbitrary amount of ADA. When executing a deposit, the following condition is checked:

```
assert(
    locked_value == (
        value.from_lovelace(lovelace_sent)
        |> value.add(own_pid, "", lovelace_sent)
    ),
    @"Recipient value must be equal to the lovelace_sent field in the redeemer!",
),
```

This means that locked_lovelace from the datum must always match the amount of ADA in the deposit.

Second, when claiming, the exact amount of ADA is checked here:

```
// previously was validating that the input was the rebated value.
// Now, we check it correctly corresponds to datum.
(( lovelace |> dict.to_list ) == [("", locked_lovelace)])?,
```

If the lovelace on the UTXO differs from the amount in the datum, the sale will be un-claimable.

Finally, when checking the result, the amount sent back to the user is compared via:

```
lovelace_amt >= must_rebate,
```

If the user had included an extra amount in their deposit to cover minUTXO, and received a rebate, then whoever executed their claim would be able to take the surplus.

Coincidentally, these issues nullify eachother: because the user cannot lock up additional ADA, there is no concern about the additional ADA being taken, or the UTXO being unspendable.

Recommendation

If you would like to preserve this property, we recommend switching the first check to comparing a greaterthan, like so:

```

assert(
  and {
    value.lovelace_of(locked_value) >= lovelace_sent,
    value.without_lovelace(locked_value) == value.add(own_pid, "", lovelace_sent)
  },
  @"Recipient value must be equal to the lovelace_sent field in the redeemer!",
),

```

Then, update the second check to compare the Counting Tokens instead, like so:

```
(( other_dict |> dict.to_list ) == [("", locked_lovelace)])?,
```

This is equivalent to the intention, since the minting policy checks that Counting Tokens are minted equal to the amount in the datum, but avoids the surplus ADA issue.

Finally, we recommend checking that the amount sent to the user is exactly:

```
lovelace_amt == this_value.lovelace - must_send
```

Which ensures that the user receives the full amount of ADA they put in, minus the amount they paid in the sale.

Alternatively, you can simply acknowledge the risk that Butane may need to cover the minUTXO costs to fully realize the claim. The upper-bound for this cost would be if the full amount was subscribed exactly (meaning no rebates were available to cover the minUTXO), and the most users possible claimed at the minimum amount available.

That would be roughly 32,250 users, with a minUTXO requirement of around 1.2 ADA, or around 38,700 ADA in minUTXO costs. In any case, the minUTXO cost is capped at 0.4% of the total raise.

Resolution

This issue was acknowledged by the project team with the comment: We will take the latter approach; If we raise 10,000,000 ADA, having to cover up to 38,700 ADA in minUTXO costs is an acceptable risk for us.

5 - Appendix

5.a - Disclaimer

This Smart Contract Security Audit Report (“Report”) is provided on an “as is” basis, for informational purposes only, and should not be construed as investment advice or any other kind of advice on legal, financial, or other matters. The entities and individuals involved in preparing this Report (“Auditors”) do not guarantee the accuracy, completeness, or usefulness of the information provided herein and shall not be held liable for any contents, errors, omissions, or inaccuracies in this Report or for any actions taken in reliance thereon.

The Auditors make no claims, promises, or guarantees about the absolute security of the smart contracts audited and the underlying code. The findings, interpretations, and conclusions presented in this Report are based on the best efforts of the Auditors and reflect their professional judgment at the time of the audit. The blockchain and cryptocurrency landscape is rapidly evolving, and new vulnerabilities may emerge that were not identified or considered at the time of the audit. As such, this Report should not be considered as a comprehensive guarantee of the audited smart contracts’ security.

The Auditors disclaim, to the fullest extent permitted by law, any and all warranties, whether express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement. The Auditors shall not be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this Report, even if advised of the possibility of such damage.

This Report is not exhaustive and is subject to change without notice. The Auditors reserve the right to update, modify, or revise this Report based on new information, subsequent developments, or further analysis. The Auditors encourage all interested parties to conduct their own independent research and due diligence when evaluating the security of smart contracts.

By using or relying on this Report, you agree to indemnify and hold harmless the Auditors from any claim, demand, action, damage, loss, cost, or expense, including attorney fees, arising out of or relating to your use of or reliance on this Report.

If you have any questions or require further clarification regarding this Report, please contact the contact@sundaeswap.finance.

5.b - Issue Guide

5.b.i - Severity

Severity	Description
Critical	Critical issues highlight exploits, bugs, loss of funds, or other vulnerabilities that prevent the dApp from working as intended. These issues have no workaround.
Major	Major issues highlight exploits, bugs, or other vulnerabilities that cause unexpected transaction failures or may be used to trick general users of the dApp. dApps with Major issues may still be functional.
Minor	Minor issues highlight edge cases where a user can purposefully use the dApp in a non-incentivized way and often lead to a disadvantage for the user.
Info	Info are not issues. These are just pieces of information that are beneficial to the dApp creator, or should be kept in mind for the off-chain code or end user. These are not necessarily acted on or have a resolution, they are logged for the completeness of the audit.
Witness	Witness findings are affirmative findings, which covers bizarre corner cases we considered and found to be safe. Not all such cases are covered, but when something is considered interesting, or might be a common question, we try to include it.

5.b.ii - Status

Status	Description
Resolved	Issues that have been fixed by the project team.
Mitigated	Issues that have a partial mitigation , and are now vulnerable in only extreme corner cases.

Acknowledged

Issues that have been **acknowledged** or **partially fixed** by the **project** team.
Projects can decide to not **fix** issues for whatever reason.

Identified

Issues that have been **identified** by the **audit** team. These are waiting for a response from the **project** team.

5.c - Revisions

This report was created using a git based workflow. All changes are tracked in a github repo and the report is produced using [typst](#). The report source is available [here](#). All versions with downloadable PDFs can be found on the [releases page](#).

5.d - About Us

Sundae Labs stands at the forefront of innovation within the Cardano ecosystem, distinguished by its pioneering development of the first Automated Market Maker (AMM) Decentralized Exchange (DEX) on Cardano. As a trusted leader in blockchain technology, we offer a comprehensive suite of products and services designed to enhance the Cardano network's functionality and security. Our offerings include Sundae Rewards, Sundae Governance, Sundae Exchange, and Sundae Taste Test—an automated price discovery platform—all available on a Software as a Service (SaaS) basis. These solutions empower other high-profile projects within the ecosystem by providing them with turnkey rewards and governance capabilities, thereby fostering a more robust and scalable blockchain infrastructure.

Beyond our product offerings, Sundae Labs is deeply committed to the advancement of the Cardano community and its underlying technology. We contribute significantly to research and development efforts aimed at improving Cardano's security and scalability. Our engagement with Input Output Global (IOG) initiatives, such as Voltaire, and participation in core technological discussions underscore our dedication to the Cardano ecosystem's growth. Additionally, our expertise extends to software development consulting services, including product design and development, and conducting security audits. Sundae Labs is not just a contributor but a vital partner in Cardano's journey towards achieving its full potential.

5.d.i - Links

Mehen Stablecoin - <https://mehen.io>

Sundae Labs - <https://sundae.fi>

Sundae Public Audits - <https://github.com/SundaeSwap-finance/sundae-audits-public>