Treasury & Voting Smart Contracts for DAO Governance × Atala PRISM

— Testing Report —

MuesliSwapTeam

April 23, 2024

1 Purpose & Scope

The purpose of this document is to provide a detailed tesing report of a set of treasury and voting/governance smart contracts for the *DAO Governance × Atala PRISM* project supported by Catalyst Fund 10. This includes the testing of all key functionalities, integration with the Cardano blockchain, and performance under different network conditions.

A draft version of the platform has been developed in OpShin, a domain-specific language for smart contracts on the Cardano blockchain based on Python. This ensure maximum maintainability and readability of the code. Nontheless, before the platform is deployed on the Cardano blockchain, it is crucial to conduct comprehensive testing to ensure the security, reliability, and efficiency of the platform. In a first step, a seperated part of the team has preformed an independent tests of the code which is presented in this document. The document is made availabile to the public to ensure transparency and to provide a comprehensive overview of the testing process and its outcomes. The document is purely informative and does not constitute financial or investment advice nor does it promise or guarantee the absence of bugs or complete faithfulness of the analyzed code.

The scope of this document entails a list of performed tests, methodologies applied, and detailed results for each tested component/property of the smart contract suite.

2 Basic Tests

Basic tests are performed to ensure the initial functionality and integrity of the smart contracts before detailed feature-specific testing. As implicit preliminary steps, this also includes compiling the contracts and deploying them on testnet. Then, as described in this section, we are running some initial checks to verify that the contracts are behaving as expected when being interacted with in the intended manner (tests for ensuring correct behavior on unintended interaction follow in the next section).

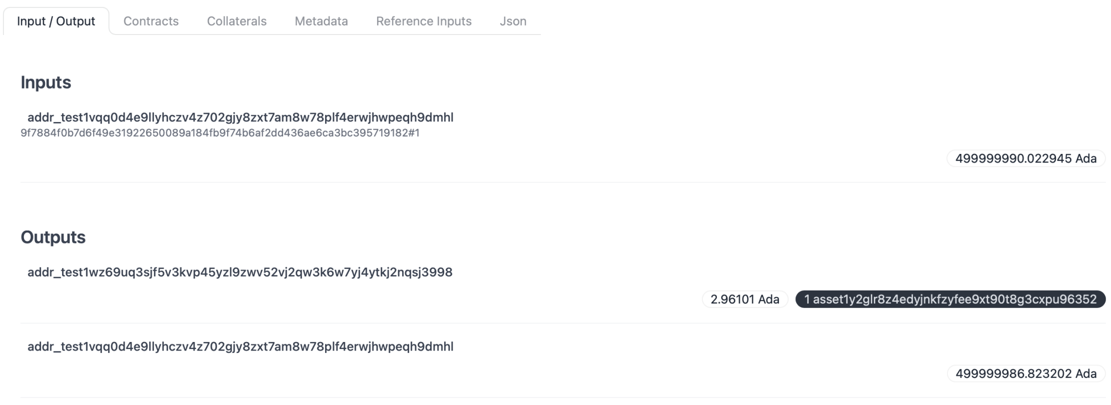
2.1 Creation of a new governance thread

Description. First, we want to ensure that a new governance thread can be successfully initialized. This will create a thread represented by a UTxO with datum containing the thread's parameters at the governance contract. As a result, the unique thread ID will be printed by our off-chain code.

Testing Procedure. We run the following script that is part of our off-chain toolset.

```
$ python3 -m muesliswap_onchain_governance.offchain.gov_state.init --wallet creator
```

Result. In the transaction below, we see the respective governance thread being created as output, containing the unique NFT that authenticates this thread.



Inputs

addr_test1vqq0d4e9llyhczv4z702gjy8zxt7am8w78plf4erwjhwpeqh9dmhl
9f7884f0b7d6f49e31922650089a184fb9f74b6af2dd436ae6ca3bc395719182#1

499999990.022945 Ada

Outputs

addr_test1wz69uq3sjf5v3kvp45yzl9zwv52vj2qw3k6w7yj4ytkj2nqsj3998

2.96101 Ada   1 asset1y2glr8z4edyjnkfzyfee9xt90t8g3cxpu96352

addr_test1vqq0d4e9llyhczv4z702gjy8zxt7am8w78plf4erwjhwpeqh9dmhl

499999986.823202 Ada

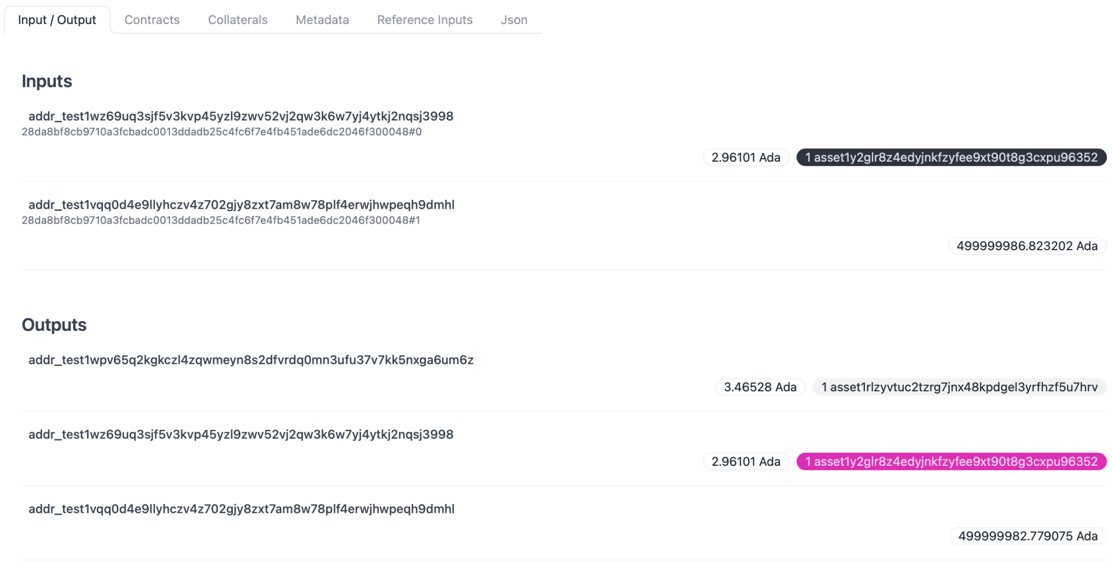
2.2 Creation of a new tally

Description. Next, within the initialized governance thread, we create a new tally. Per default, this tally will be open for 10 minutes and allow staked users to add their votes which will be written into the tallies datum.

Testing Procedure. We run the following script that is part of our off-chain toolset.

```
$ python3 -m muesliswap_onchain_governance.offchain.gov_state.create_tally \
--wallet creator
```

Result. In the transaction below, we see the governance thread created in the previous transaction being provided as input. As output we again have this governance thread, plus the newly created tally which is authenticated by another unique tally NFT.



Inputs

addr_test1wz69uq3sjf5v3kvp45yzl9zwv52vj2qw3k6w7yj4ytkj2nqsj3998
28da8bf8cb9710a3fcbadc0013ddadb25c4fc6f7e4fb451ade6dc2046f300048#0
2.96101 Ada | 1 asset1y2glr8z4edyjnkfzyfee9xt90t8g3cxpu96352

addr_test1vqq0d4e9llyhczv4z702gjy8zxt7am8w78plf4erwjhwpeqh9dmhl
28da8bf8cb9710a3fcbadc0013ddadb25c4fc6f7e4fb451ade6dc2046f300048#1
499999986.823202 Ada

Outputs

addr_test1wpv65q2kgkczl4zqwmeyn8s2dfvrdq0mn3ufu37v7kk5nxga6um6z
3.46528 Ada | 1 asset1rlzyvtuc2tzrg7jnx48kpdgel3yrfhzf5u7hrv

addr_test1wz69uq3sjf5v3kvp45yzl9zwv52vj2qw3k6w7yj4ytkj2nqsj3998
2.96101 Ada | 1 asset1y2glr8z4edyjnkfzyfee9xt90t8g3cxpu96352

addr_test1vqq0d4e9llyhczv4z702gjy8zxt7am8w78plf4erwjhwpeqh9dmhl
499999982.779075 Ada

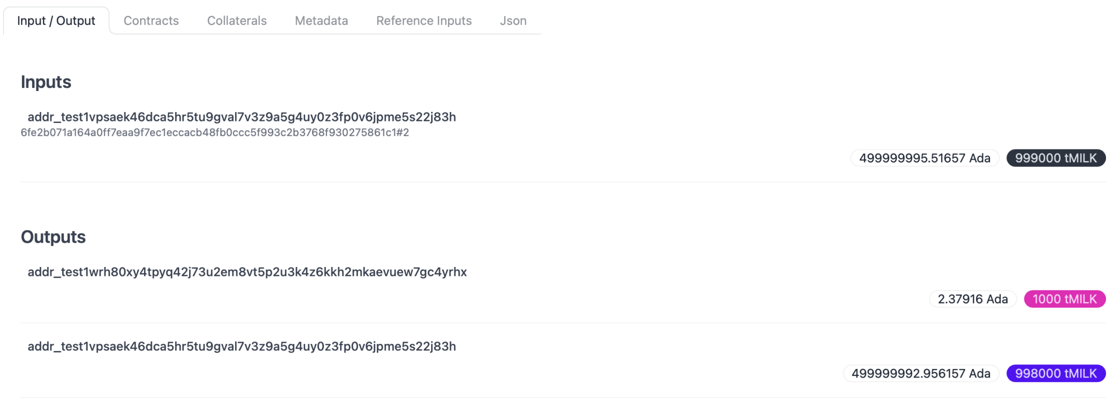
2.3 Creation of a new staking position

Description. Next, we want to move towards voting in the above created tally. For this we first need to have the voter stake some of her governance tokens into the dedicated staking contract. This is tested in the following.

Testing Procedure. We run the following script that is part of our off-chain toolset.

```
$ python3 -m muesliswap_onchain_governance.offchain.staking.init --wallet voter
```

Result. In the transaction below, we see how the voter locks 1000 tMILK tokens into the staking contract (first output) while her remaining tokens are returned to the voter's wallet (second output).



Inputs

addr_test1vpsaek46dca5hr5tu9gval7v3z9a5g4uy0z3fp0v6jpme5s22j83h
6fe2b071a164a0ff7eaa9f7ec1eccacb48fb0ccc5f993c2b3768f930275861c1#2
499999995.51657 Ada | 999000 tMILK

Outputs

addr_test1wrh80xy4tpyq42j73u2em8vt5p2u3k4z6kkh2mkaevuew7gc4yrhx
2.37916 Ada | 1000 tMILK

addr_test1vpsaek46dca5hr5tu9gval7v3z9a5g4uy0z3fp0v6jpme5s22j83h
499999992.956157 Ada | 998000 tMILK

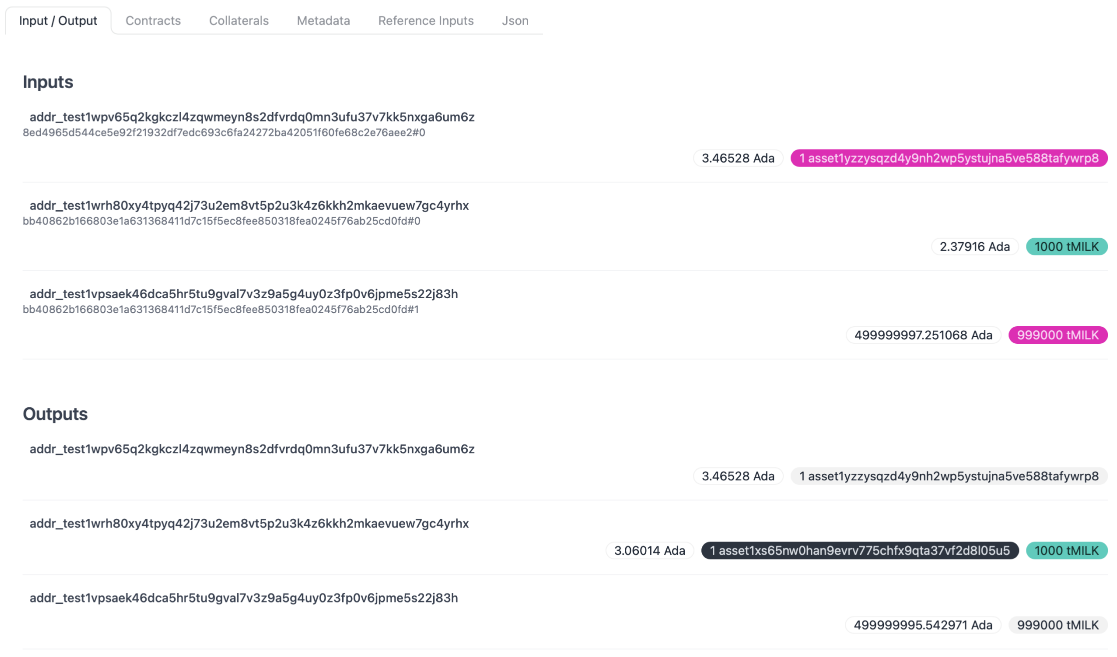
2.4 Adding a vote to a tally

Description. Next, we want to actually have the user vote. The tally contract will check the voter's stake (here locked as tMILK into the staking contract) and enter a vote with the respective weight into the tally contract.

Testing Procedure. We run the following script that is part of our off-chain toolset.

```
$ python3 -m muesliswap_onchain_governance.offchain.tally.add_vote_tally \
--wallet voter --proposal_id 1 --proposal_index 1
```

Result. Here, as inputs to the transaction, we have the previously created tally, as well as the UTxO containing the staked tMILK. The transactions outputs are given by the same UTxOs, where the staking UTxO now has an additional token representing the added vote.



Inputs

addr_test1wpv65q2kgkczl4zqwmeyn8s2dfvrdq0mn3ufu37v7kk5nxga6um6z
8ed4965d544ce5e92f21932df7edc693c6fa24272ba42051f60fe68c2e76aee2#0
3.46528 Ada | 1 asset1yzzysqzd4y9nh2wp5ystujna5ve588tafywrp8

addr_test1wrh80xy4tpyq42j73u2em8vt5p2u3k4z6kkh2mkaevuew7gc4yrhx
bb40862b166803e1a631368411d7c15f5ec8fee850318fea0245f76ab25cd0fd#0
2.37916 Ada | 1000 tMILK

addr_test1vpsaek46dca5hr5tu9gval7v3z9a5g4uy0z3fp0v6jpme5s22j83h
bb40862b166803e1a631368411d7c15f5ec8fee850318fea0245f76ab25cd0fd#1
499999997.251068 Ada | 999000 tMILK

Outputs

addr_test1wpv65q2kgkczl4zqwmeyn8s2dfvrdq0mn3ufu37v7kk5nxga6um6z
3.46528 Ada | 1 asset1yzzysqzd4y9nh2wp5ystujna5ve588tafywrp8

addr_test1wrh80xy4tpyq42j73u2em8vt5p2u3k4z6kkh2mkaevuew7gc4yrhx
3.06014 Ada | 1 asset1xs65nw0han9evrv775chfx9qta37vf2d8l05u5 | 1000 tMILK

addr_test1vpsaek46dca5hr5tu9gval7v3z9a5g4uy0z3fp0v6jpme5s22j83h
499999995.542971 Ada | 999000 tMILK

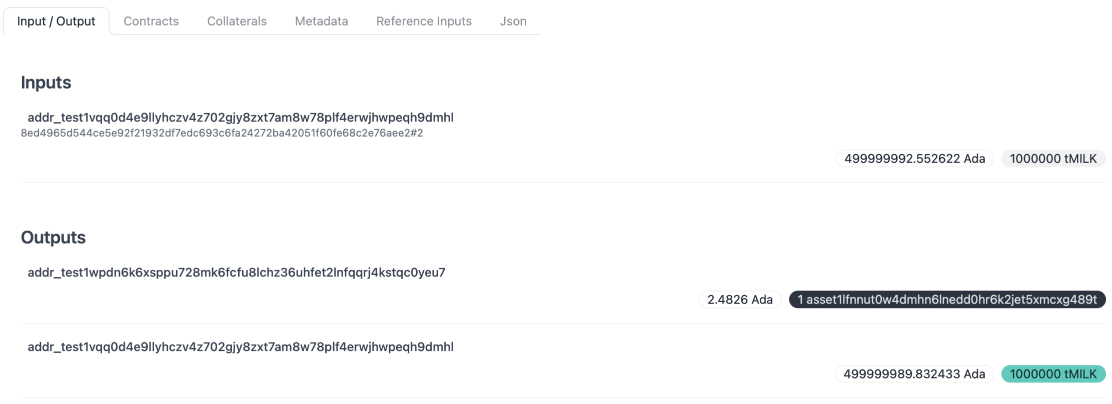
2.5 Initialization of the treasury

Description. In order to test whether positive election outcomes can successfully be used to initiate the resulting actions, here in terms of treasury payouts, we first need to initialize a treasury. This is done by creating an initial UTxO at the treasury contract with the respective treasury parameters in the datum.

Testing Procedure. We run the following script that is part of our off-chain toolset.

```
$ python3 -m muesliswap_onchain_governance.offchain.treasury.init --wallet creator
```

Result. Observe that a treasury output UTxO is created, containing by a unique treasury authentication NFT.



Input / Output | Contracts | Collaterals | Metadata | Reference Inputs | Json

Inputs

addr_test1vqq0d4e9llyhczv4z702gjy8zxt7am8w78plf4erwjhwpeqh9dmhl
8ed4965d544ce5e92f21932df7edc693c6fa24272ba42051f60fe68c2e76aee2#2

499999992.552622 Ada | 1000000 tMILK

Outputs

addr_test1wpdn6k6xsppu728mk6fcfu8lchz36uhfet2lnfqqrj4kstqc0yeu7

2.4826 Ada | 1 asset1lfnnut0w4dmhn6lnedd0hr6k2jet5xmcxg489t

addr_test1vqq0d4e9llyhczv4z702gjy8zxt7am8w78plf4erwjhwpeqh9dmhl

499999989.832433 Ada | 1000000 tMILK

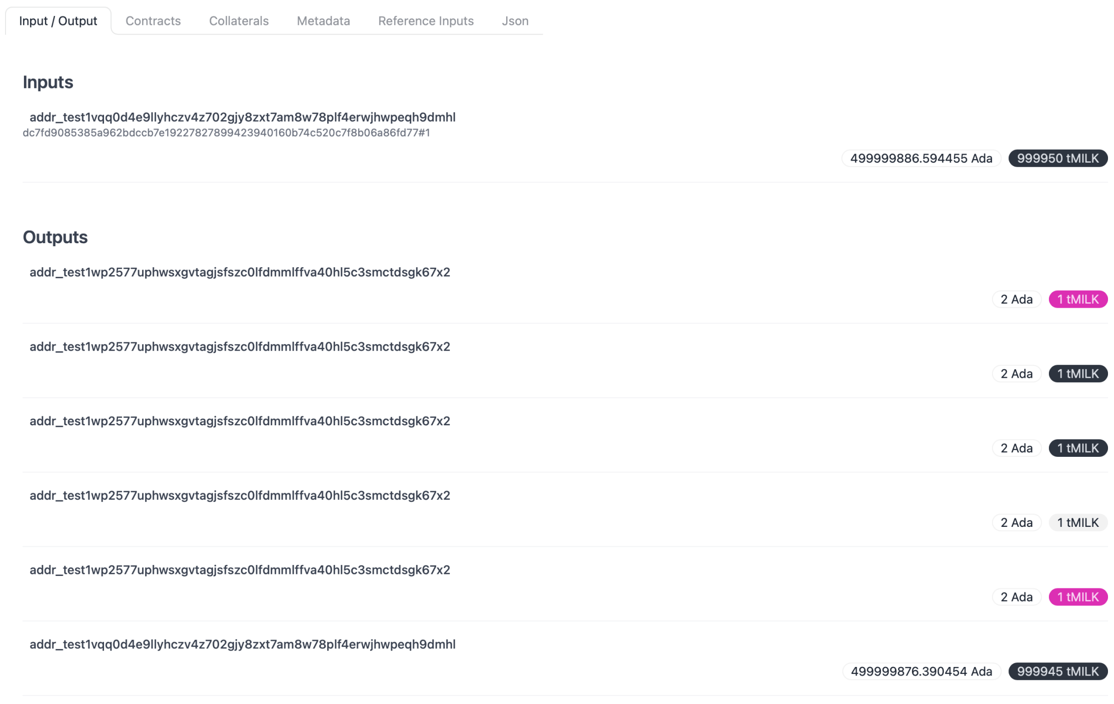
2.6 Deposit funds to treasury

Description. Next, we need to deposit some funds into the above initialized treasury UTxO. This is done as follows.

Testing Procedure. We run the following script that is part of our off-chain toolset.

```
$ python3 -m muesliswap_onchain_governance.offchain.treasury.deposit --wallet creator
```

Result. As outputs, in addition to the UTxO with the treasury authentication NFT, we get 5 new UTxOs at the treasury's value store contract holding the actual treasury funds. The desired number of such value store outputs can be configured in the script above.



Inputs

addr_test1vqq0d4e9llyhczv4z702gjy8zxt7am8w78plf4erwjhwpeqh9dmhl
dc7fd9085385a962bdccb7e19227827899423940160b74c520c7f8b06a86fd77#1
499999886.594455 Ada 999950 tMILK

Outputs

addr_test1wp2577uphwsxgvtagjsfszc0lfdmmlffva40hl5c3smctdsgk67x2
2 Ada 1 tMILK

addr_test1wp2577uphwsxgvtagjsfszc0lfdmmlffva40hl5c3smctdsgk67x2
2 Ada 1 tMILK

addr_test1wp2577uphwsxgvtagjsfszc0lfdmmlffva40hl5c3smctdsgk67x2
2 Ada 1 tMILK

addr_test1wp2577uphwsxgvtagjsfszc0lfdmmlffva40hl5c3smctdsgk67x2
2 Ada 1 tMILK

addr_test1wp2577uphwsxgvtagjsfszc0lfdmmlffva40hl5c3smctdsgk67x2
2 Ada 1 tMILK

addr_test1vqq0d4e9llyhczv4z702gjy8zxt7am8w78plf4erwjhwpeqh9dmhl
499999876.390454 Ada 999945 tMILK

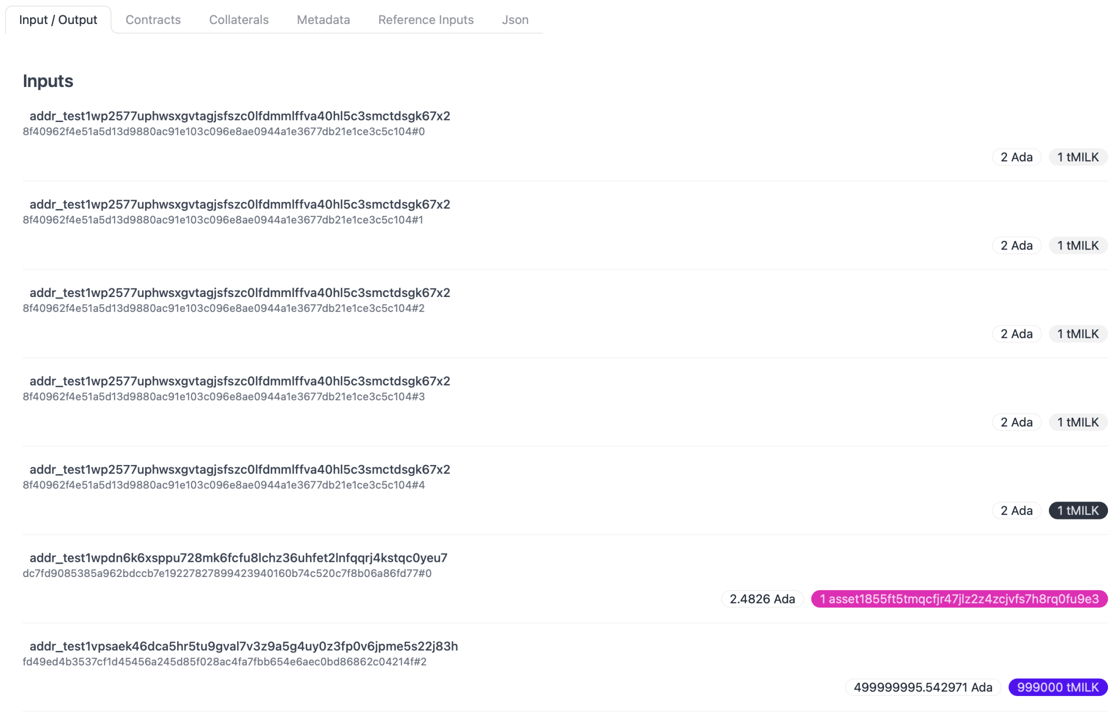
2.7 Treasury payout (after successful vote)

Description. Finally, we want to use the outcome of our above created test election (which was about allowing a payout to the voter wallet, and which received one positive vote) in order to conduct the actual payout from the treasury.

Testing Procedure. We run the following script that is part of our off-chain toolset.

```
$ python3 -m muesliswap_onchain_governance.offchain.treasury.payout --wallet voter
```

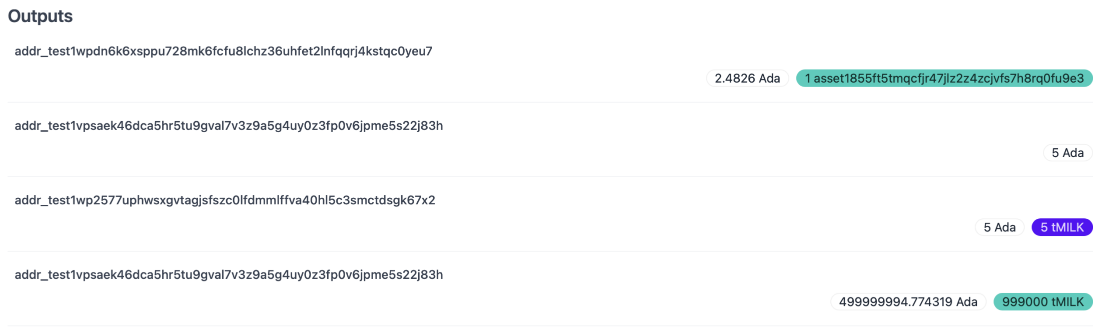
Result. As inputs, we have the treasury's value store UTxOs and the UTxO representing the tally state which also contains the election outcome information.


Input / Output | Contracts | Collaterals | Metadata | Reference Inputs | Json

Inputs

```
addr_test1wp2577uphwsxgvtagjsfszc0lfdmmlffva40hl5c3smctdsgk67x2
8f40962f4e51a5d13d9880ac91e103c096e8ae0944a1e3677db21e1ce3c5c104#0
                                                      2 Ada   1 tMILK

addr_test1wp2577uphwsxgvtagjsfszc0lfdmmlffva40hl5c3smctdsgk67x2
8f40962f4e51a5d13d9880ac91e103c096e8ae0944a1e3677db21e1ce3c5c104#1
                                                      2 Ada   1 tMILK

addr_test1wp2577uphwsxgvtagjsfszc0lfdmmlffva40hl5c3smctdsgk67x2
8f40962f4e51a5d13d9880ac91e103c096e8ae0944a1e3677db21e1ce3c5c104#2
                                                      2 Ada   1 tMILK

addr_test1wp2577uphwsxgvtagjsfszc0lfdmmlffva40hl5c3smctdsgk67x2
8f40962f4e51a5d13d9880ac91e103c096e8ae0944a1e3677db21e1ce3c5c104#3
                                                      2 Ada   1 tMILK

addr_test1wp2577uphwsxgvtagjsfszc0lfdmmlffva40hl5c3smctdsgk67x2
8f40962f4e51a5d13d9880ac91e103c096e8ae0944a1e3677db21e1ce3c5c104#4
                                                      2 Ada   1 tMILK

addr_test1wpdn6k6xsppu728mk6fcfu8lchz36uhfet2lnfqqrj4kstqc0yeu7
dc7fd9085385a962bdccb7e19227827899423940160b74c520c7f8b06a86fd77#0
              2.4826 Ada   1 asset1855ft5tmqcfjr47jlz2z4zcjvfs7h8rq0fu9e3

addr_test1vpsaek46dca5hr5tu9gval7v3z9a5g4uy0z3fp0v6jpme5s22j83h
fd49ed4b3537cf1d45456a245d85f028ac4fa7fbb654e6aec0bd86862c04214f#2
                                  499999995.542971 Ada   999000 tMILK
```

From the outputs, we can observe that a payout of 5 Ada to the voter's wallet is taking place.


Outputs

```
addr_test1wpdn6k6xsppu728mk6fcfu8lchz36uhfet2lnfqqrj4kstqc0yeu7
              2.4826 Ada   1 asset1855ft5tmqcfjr47jlz2z4zcjvfs7h8rq0fu9e3

addr_test1vpsaek46dca5hr5tu9gval7v3z9a5g4uy0z3fp0v6jpme5s22j83h
                                                              5 Ada

addr_test1wp2577uphwsxgvtagjsfszc0lfdmmlffva40hl5c3smctdsgk67x2
                                                      5 Ada   5 tMILK

addr_test1vpsaek46dca5hr5tu9gval7v3z9a5g4uy0z3fp0v6jpme5s22j83h
                                  499999994.774319 Ada   999000 tMILK
```

3 Testing Security Properties

Security testing focuses on verifying the robustness and reliability of smart contracts against common vulnerabilities and attacks. This section details tests aimed at ensuring that election cannot be manipulated. Moreover, we descibe how users can verify that their votes are taken into account correctly.

3.1 User cannot vote twice

Description. In order to provide a fair election mechanism, it is of course essential to ensure that each user can only vote at most once on each tally. This is achieved by storing a reference to each voter's staking position (represented by the respective staking authentication NFT) in the tally's datum. In the following test we try to have a user vote twice on the same tally.

Testing Procedure. After creating a new tally and the voter's staking position as described before, we run the following commands to submit the same vote twice.

```
$ python3 -m muesliswap_onchain_governance.offchain.tally.add_vote_tally \
--wallet voter --proposal_id 1 --proposal_index 1 \
$ python3 -m muesliswap_onchain_governance.offchain.tally.add_vote_tally \
--wallet voter --proposal_id 1 --proposal_index 1
```

Result. While the first voting transaction succeeds as before, when executing it for the second time, we get the following script failure trace:

```
'validationError': "An error has occurred:  User error: The machine terminated
because of an error, either from a built-in function or from an explicit use of
'error'.",
'traces': ['Stake already participates in vote']
```

3.2 User (or 3rd party) can verify that vote has been counted

Description. As a user, it is desirable to verify whether one's vote is actually taken into account. Since the entire tally state is stored on-chain in the tally UTxO's datum, this can be checked by looking at the tally output's datum of an add-vote transaction.

Testing Procedure. This does not require constructing any new transaction, we simply look at the output datum of above add-vote transaction.

Result. Here's the tally datum before adding the vote:

```
TallyState(votes=[0, 0, 0, 0], params=ProposalParams(...))
```

And here is the datum after adding the vote:

```
TallyState(votes=[0, 1000, 0, 0], params=ProposalParams(...))
```

As one can see, the vote has been entered accordingly with weight 1000. By inspecting the contract logic, the user can convince herself that votes contained in this datum are correctly taken into account.

3.3 No treasury payout if majority not reached

Description. For this test we aim to construct a situation where no majority has been reached in a election, to demonstrate that then the proposed action cannot be executed. For this, we use the same tally as before, but in addition to the positive vote, we also add one negative vote. Then we try to perform the payout from the treasury.

Testing Procedure. After creating the tally, and staking positions for both the voter and creator wallets as shown before, we run the following commands to vote, each for a different proposal outcome. Then we try to perform the previously shown payout.

Result. As desired, we get the following script failure trace:

```
'validationError': "An error has occurred:  User error: The machine terminated
because of an error, either from a built-in function or from an explicit use of
'error'.",
'traces': ['Quorum not reached']
```

4 Conclusion

The comprehensive testing covered in this report indicates that the treasury and voting/governance smart contracts for the DAO Governance × Atala PRISM project are robust and function as expected. While no major security issues were detected for this draft version of the smart contracts, continuous re-testing and reviews are necessary to ensure the contracts are suitable for deployment in a real-world DAO. For further results on the security of the tested smart contracts, we refer to the detailed audit report which also has been provided as part of the same Catalyst-funded project.