Security Audit

SGN and SGA Smart Contracts

KANSO LABS

Leonid Beder

August 2019

VERSION 2.1

PRIVATE & CONFIDENTIAL

Table of Contents

Introduction

Kanso Labs was engaged to perform a time-boxed security review of the Saga Core's smart contracts. The review focused on security and functional aspects of the Solidity implementation of the smart contracts. General recommendations and informational comments are also provided.

The code is excellent, very straightforward, and excellently tested.

The team has followed most of our recommendations and updated the contracts. The issues that weren't fixed do not affect the safety or correctness of the models and their implementations.

This report is a follow-up on the previous "SGN and SGA Smart Contracts Security Audit" v1.0, v1.1, v1.2, and v2.0 reports. Kudos to the team for thoroughly addressing and committing to resolving previously found issues.

Please see Appendix A: Audited Files for a detailed list of the audited files and their versions.

Saga Core

About the Company

Saga is a non-profit foundation headquartered in Switzerland. It is governed by Swiss law and regulations, including the rules of the Swiss Federal Supervisory Board for Foundations (ESA), and the Financial Market Supervisory Authority (FINMA).

Saga has a growing team of experienced professionals who are developing practical solutions to fulfill the promise of a digital currency that can work within the existing financial system. The team includes experienced entrepreneurs, technology experts, researchers, economists, and financial professionals.

For more info, please visit https://www.saga.org

Saga: An Asset-Backed Currency

Saga introduces a non-sovereign, non-anonymous, and reserve backed digital currency. By allowing participants to both buy and sell SGA, Saga's smart contract acts as a market maker. To buy SGA tokens, participants send funds to Saga's smart contract, where they are kept as part of the variable reserve of conventional currencies, hosted by reputable banks.

The primary purpose of Saga's reserve is to ensure participants can sell SGA. The contract will always offer to buy SGA, drawing on funds from the reserve.

For further info, please refer to Saga's impressively elegant documentation below:

- Executive Summary: https://www.saga.org/static/files/executive-summary.pdf
- Whitepaper: https://www.saga.org/static/files/saga-whitepaper.pdf
- Monetary Model: https://www.saga.org/static/files/saga-monetary-model.pdf

Disclaimer

This report reflects our current understanding of known security patterns as they relate to the Saga Core contracts. It is not an endorsement of the reliability or effectiveness of the contracts, merely an assessment of their logic and robustness. Kanso Labs has not reviewed the related Saga Core project, nor its backend administration and operation system, deployment scripts, and operational or organizational security.

This report should not be viewed as an investment or legal advice. It is provided by Kanso Labs as is, without warranty of any kind, express or implied. In no event shall Kanso Labs be liable concerning any subject matter relating to this report including any decisions based upon it.

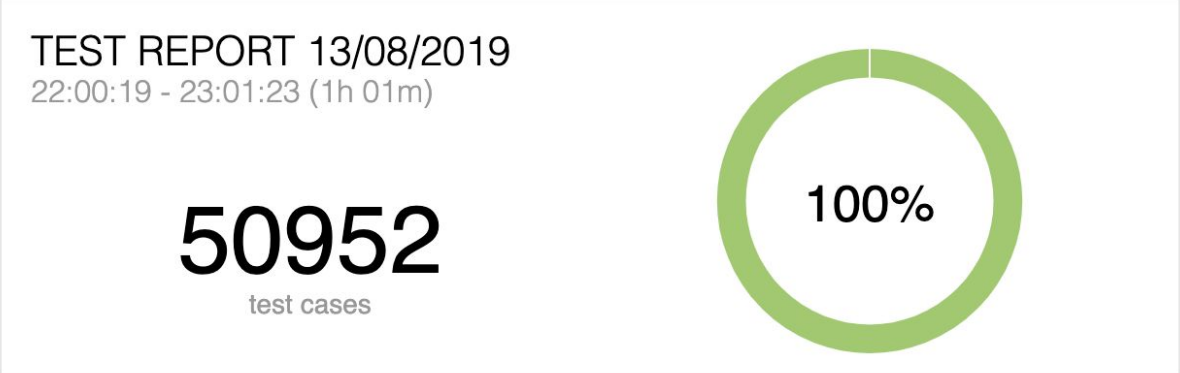
Formal security audits are not enough to guarantee secure smart contracts. Kanso Labs recommends that Saga Core establish a bug bounty program, setting a period during which security researchers attempt to break the token's invariants, in turn encouraging further analysis of the contract.

Given the size of the project, the findings detailed here are not to be considered exhaustive, and further testing and audit are recommended after the issues covered are fixed.

Testing

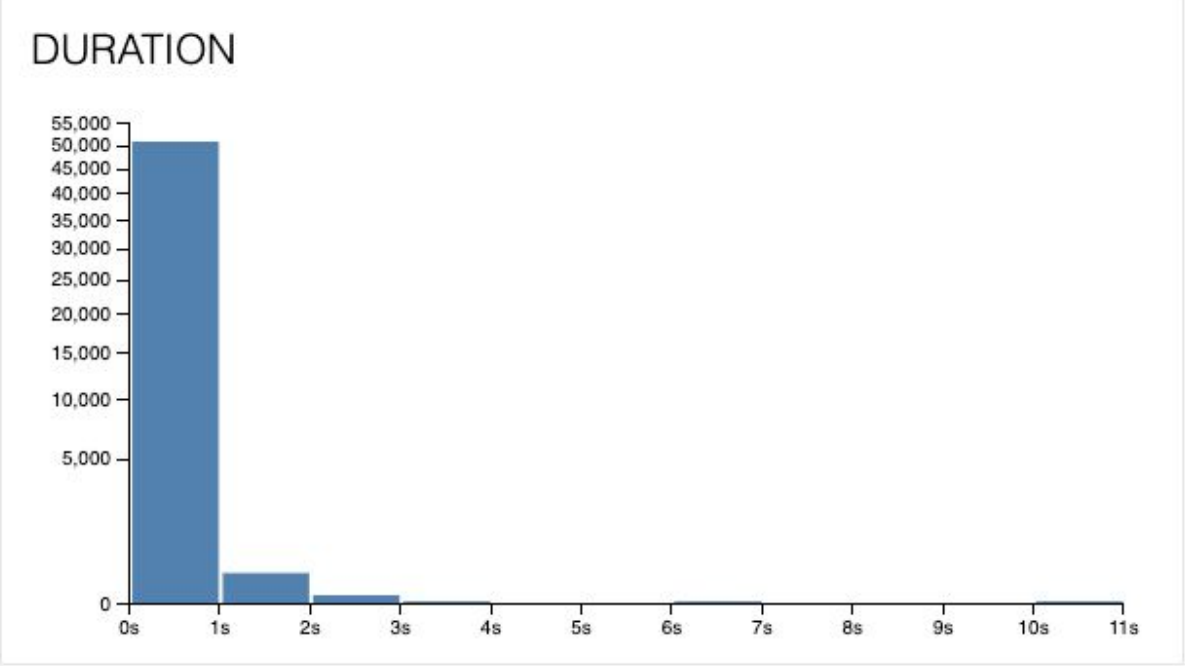
Analysis

Congratulations on writing such a thorough test suite! There are **50,952** tests in total, including an extensive test matrix, with most of them covering the integration of all contracts into the SGN, and the SGA contracts:



Well done!

Most of the test cases were executed pretty quickly:



Please refer to Appendix C: Test Cases for a complete list of test cases.

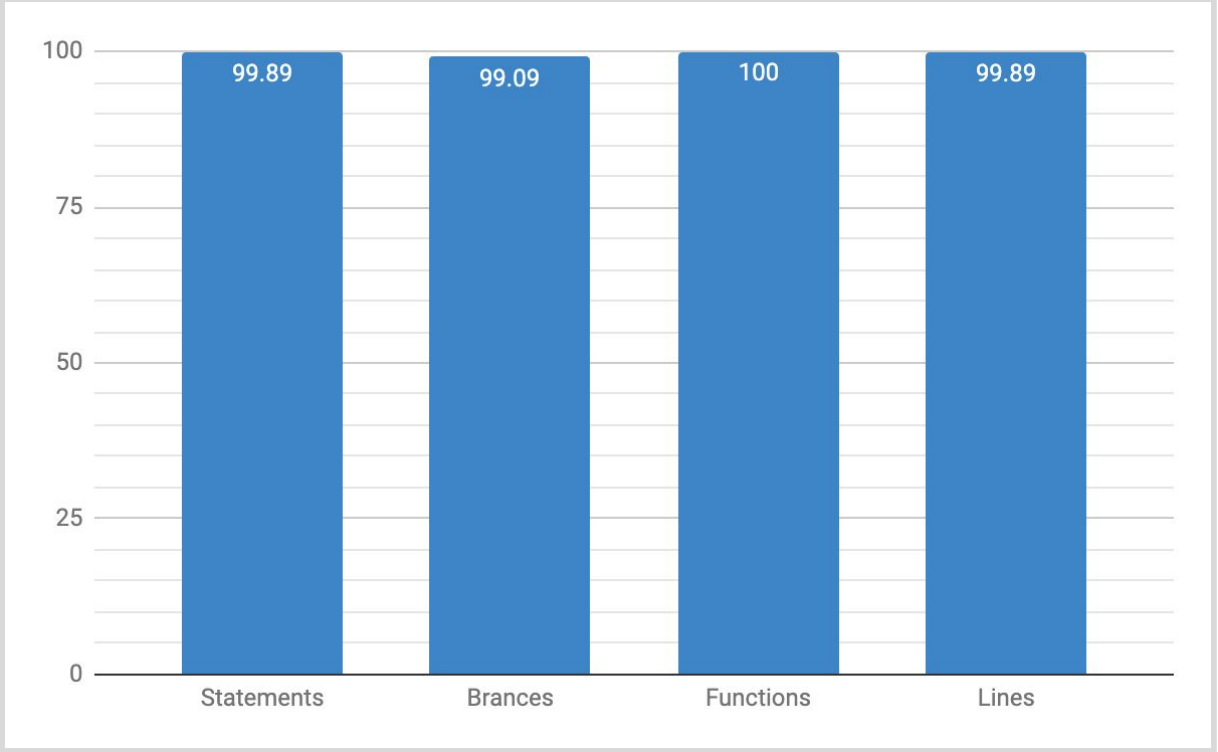
Code Coverage

The following table shows the test code coverage for this version of the Saga Core smart contracts:

99.89% Statements 903/904 **99.09%** Branches 325/328 **100%** Functions 265/265 **99.89%** Lines 869/870

File		Statements		Branches		Functions		Lines	
authorization/		100%	38/38	100%	10/10	100%	18/18	100%	28/28
authorization/interfaces/		100%	0/0	100%	0/0	100%	0/0	100%	0/0
contract_address_locator/		100%	27/27	100%	16/16	100%	12/12	100%	29/29
contract_address_locator/interfaces/		100%	0/0	100%	0/0	100%	0/0	100%	0/0
saga-genesis/		100%	178/178	100%	24/24	100%	33/33	100%	180/180
saga-genesis/interfaces/		100%	0/0	100%	0/0	100%	0/0	100%	0/0
saga/		100%	507/507	100%	200/200	100%	153/153	100%	459/459
saga/interfaces/		100%	0/0	100%	0/0	100%	0/0	100%	0/0
utils/		99.03%	102/103	94.44%	51/54	100%	32/32	99.18%	121/122
wallet_trading_limiter/		100%	51/51	100%	24/24	100%	17/17	100%	52/52
wallet_trading_limiter/interfaces/		100%	0/0	100%	0/0	100%	0/0	100%	0/0

The total code coverage of **99.89%** statements, **99.09%** branches, **100%** functions, and **99.89%** lines is very high and in full compliance with the standards of the industry.



The only partially covered functional contract is **MultiSigWallet.sol** (**98.84%** statements, **93.18%** branches, **100%** functions, **99.02%** lines), which is the de-facto standard Gnosis MultiSigWallet smart contract[1].

[1] We'd encourage the team to bring MultiSigWallet's code coverage to a perfect 100% as well and will be more than happy to provide the team with our own test suite, which already achieves that.

Simulation

As part of the testing of the monetary model, we've implemented a simulation of the model in Rust, which can be found here https://github.com/lbeder/saga-model-simulator.

We have used the simulator to:

1. Verify the original Saga scenarios in **test/saga/helpers/sequences**. The original scenarios, in the supported JSON format, can be found here https://github.com/lbeder/saga-model-simulator/tree/master/scenarios/saga
2. Generate additional scenarios matrix and run it against Saga's smart contracts. The generated scenarios can be found here https://github.com/lbeder/saga-model-simulator/tree/master/scenarios/generated

Usage

```
Usage: saga-model-simulator [options]
Version: 0.0.5

Options:
      --generate      set to generate a simulation scenario
      --count COUNT   number of orders to generate
      --output OUTPUT generation output
      --verify        set to verify a simulation scenario
      --input INPUT   verification input
      --high_price_n HIGH_PRICE_N
                      numerator of the SDR/ETH high price (default: 1)
      --high_price_d HIGH_PRICE_D
                      denominator of the SDR/ETH high price (default: 1)
      --low_price_n LOW_PRICE_N
                      numerator of the SDR/ETH low price (default: 1)
      --low_price_d LOW_PRICE_D
                      denominator of the SDR/ETH low price (default: 1)
      --factor_n FACTOR_D
                      numerator of the SDR adjustment factor (default: 1)
      --factor_d FACTOR_N
                      denominator of the SDR adjustment factor (default: 1)
  -h, --help          print this help menu
```

File Format

Scenarios are provided using the following JSON format:


```
[
 {
  "func": "buy",
  "input": "28777529211672147109811790 40",
  "output": "10734300266250479780839235 22",
  "elapsed": 58059
 },
 {
  "func": "sell",
  "input": "157517002939499020825372 0718",
  "output": "75880864380658880930566818 71",
  "elapsed": 39228
 },
 .
 .
 .
]
```


Verification

To verify a scenario against the simulator, you'd need to execute:

```
./target/release/saga-model-simulator[2] \
--verify --output --input [PATH_TO_INPUT_JSON] \
  --high_price_n [HIGH_PRICE_N] --high_price_d [HIGH_PRICE_D] \
  --low_price_n [LOW_PRICE_N] --low_price_d [LOW_PRICE_D] \
  --factor_n [FACTOR_N] --factor_d [FACTOR_D]
```

To verify a scenario against Saga's smart contracts, you'd need to replace the existing js files in **test/saga/helpers/sequences** with a separate js file and replace the value of the **module.exports.sequence** export with the JSON contents of the scenario file and finally run **npm test**.

Please note that you'd need to disable the tests at **test/saga/SystemTest.js**, as the simulation scenario doesn't generate minting points verification data, which is required by it.

[2] Alternatively, you can build and run it directly via **cargo run --release --**

Generation

To generate a random scenario file, you'd need to execute:

```
./target/release/saga-model-simulator \
  --generate --output [PATH_TO_OUTPUT_JSON] --count [COUNT] \
  --high_price_n [HIGH_PRICE_N] --high_price_d [HIGH_PRICE_D] \
  --low_price_n [LOW_PRICE_N] --low_price_d [LOW_PRICE_D] \
  --factor_n [FACTOR_N] --factor_d [FACTOR_D]
```

Please make sure that $\frac{high\_price\_d}{high\_price\_n} \geq \frac{low\_price\_d}{low\_price\_n}$.

Scenarios

The generated scenarios were generated according to the following matrix:

File Name	Steps	high_price_n	high_price_d	low_price_n	low_price_d	factor_n	factor_d
hn=1,hd=1,ln=1,ld=1,fn=1,fd=1.json	1000	1	1	1	1	1	1[3]
hn=1,hd=1,ln=1,ld=1,fn=1,fd=2.json	1000	1	1	1	1	1	2
hn=1,hd=1,ln=1,ld=1,fn=2,fd=1.json	1000	1	1	1	1	2	1
hn=1,hd=1,ln=1,ld=2,fn=1,fd=1.json	1000	1	1	1	2	1	1
hn=1,hd=2,ln=1,ld=2,fn=1,fd=1.json	1000	1	2	1	2	1	1
hn=2,hd=1,ln=1,ld=1,fn=1,fd=1.json	1000	2	1	1	1	1	1
hn=2,hd=1,ln=2,ld=1,fn=1,fd=1.json	1000	2	1	2	1	1	1

All the scenarios were run against Saga's smart contracts and passed successfully.

[3] Please note that the original Saga scenarios were only tested against the default case.

Findings and Recommendations

Described below are the findings and recommendations for the audited version[4].

Kudos to the team of their exceptional architecture, productive utilization of interfaces, and precise setting of function visibility in all of their code. The code is very modular and looks thoughtfully designed.

Please note that many or development recommendations are only referenced once, while may repeat in other places in the code. In contrast, we tried to reference all appearances of security findings explicitly, but some could have been omitted.

[4] Please see Appendix A: Audited Files for a detailed list of the audited files and their versions

Mechanism

Denial of Service

Unbound Loop in ContractAddressLocator

ContractLocator's constructor receives two lists (a **bytes32[]** array of interface names and their respective **address[]** array of addresses):

In **contracts/contract_address_locator/ContractAddressLocator.sol:**

```
29      constructor(bytes32[] memory _identifiers, address[] _contractAddresses) public {
30          identifiersCount = _identifiers.length;
31          require(identifiersCount == _contractAddresses.length, "list lengths are not equal");
32          for (uint256 i = 0; i < identifiersCount; i++) {
33              require(uint256(contractAddresses[_identifiers[i]]) == 0, "identifiers are not unique");
34              contractAddresses[_identifiers[i]] = _contractAddresses[i];
35              emit Mapped(_identifiers[i], _contractAddresses[i]);
36          }
37      }
```

If constructed with large enough lists, the construction's gas cost can exceed the block gas limit. Even though this is mostly administrative contract, we'd recommend restricting these arrays with an upper bound (e.g., **MAX_INTERFACES** of 100) with an accompanying test case whenever it's utilized (e.g., testing **ContractAddressLocatorProxy** against a **ContractLocator** with predefined **MAX_INTERFACES** interfaces in order to make sure that additional operations don't exceed the block gas limit).

As a last resort, we would recommend to explicitly document that since it's an administrative and controlled method, gas limit exceeding is acceptable and won't cause DoS (other than wasting some gas).

UPDATE: The team has updated the documentation:

Added the following documentation in the `ContractAddressLocator` contract:

This contract is designated to be deployed every time the system is upgraded. Deployment will fail if the length of the lists yields gas requirement larger than the block gas-limit. However, there is no point in setting a restriction on the length of the lists in order to prevent this scenario. Instead, if such a scenario is ever encountered, this function will need to be adjusted in order to allow its execution.

Fallback Function Consumes More than 2,300 Gas

The fallback function, in **contracts/saga/SGAToken.sol** consumes more than 2,300 gas:

```
49      function() external payable {
50          uint256 amount = getSGATokenManager().exchangeEthForSga(msg.sender, msg.value);
51          _mint(msg.sender, amount);
52      }
```

In the result, if another smart contract would want to interact with **SGAToken** by sending Ether (i.e., using the **address**'s **transfer** method) - the operation will exceed the 2,300 gas stipend and fail. Please note, that externally sending Ether (or calling the fallback function), should work as intended (assuming the supplied gas limit is sufficient).

We would recommend to either document this behavior or avoid having a fallback function altogether (as it's possible to achieve the same behavior by calling the **exchange** method, which is a duplication of the same code).

UPDATE: The team has updated the documentation:

Added the following documentation in the `SGAToken` contract:

- *The fallback function: Can be executed from externally-owned accounts but not from other contracts, due to the insufficient gas-stipend provided to the fallback function.*
- *The `exchange` function: Can be executed from externally-owned accounts as well as from other contracts.*

Timeout Manipulation By Miners

In **contracts/saga/MintingPointTimersManager.sol**, timer expiration is defined by the **timeout** state variable, as set in the constructor:

```
10  contract MintingPointTimersManager is IMintingPointTimersManager, ContractAddressLocatorHolder {
11      string public constant VERSION = "1.0.0";
12
13      using SafeMath for uint256;
14
15      struct Timestamp {
16          bool valid;
17          uint256 value;
18      }
19
20      uint256 public timeout;
21      Timestamp[105] public timestamps;
22
23      /**
24       * @dev Create the contract.
25       * @param _contractAddressLocator The contract address locator.
26       * @param _timeout The number of seconds elapsed between 'running' and 'expired'.
27       * @notice Each timestamp can be in either one of 3 states: 'running', 'expired', 'neither'.
28       */
29      constructor(IContractAddressLocator _contractAddressLocator, uint256 _timeout) ContractAddressLocatorHolder(_contrac
30          timeout = _timeout;
31      }
32
```

```
63      function running(uint256 _id) external view returns (bool) {
64          Timestamp storage timestamp = timestamps[_id];
65          if (!timestamp.valid)
66              return false;
67          return timestamp.value.add(timeout) >= time();
68      }
69
70      /**
71       * @dev Get an indication of whether or not a given timestamp is 'expired'.
72       * @param _id The ID of the timestamp.
73       * @return An indication of whether or not a given timestamp is 'expired'.
74       * @notice Even if this timestamp is not 'expired', it is not necessarily 'running'.
75       */
76      function expired(uint256 _id) external view returns (bool) {
77          Timestamp storage timestamp = timestamps[_id];
78          if (!timestamp.valid)
79              return false;
80          return timestamp.value.add(timeout) < time();
81      }
```

Since miners can try to postpone transactions, which rely on **MintingPointTimersManager**, we recommend enforcing a minimum timeout duration which conforms to both accepted delay and reorg attacks, e.g., as a multiple of Ethereum median block time (~ 16 seconds).

UPDATE: The team has updated that they are going to use much longer timeouts in productions (which can be confirmed by their management scripts), thus greatly reducing the chances for miners interference:

We are planning to deploy the `MintingPointTimersManager` contract with a timeout of 1 week (as stated in the Monetary Model section on white-paper. This value is not reconfigurable during runtime (set only in the constructor of the contract) and visible to anyone who wishes to see it. For testing purposes, we prefer to keep it configurable during deployment (via the constructor).

The alternative of implementing a `timeout` function which returns a hard-coded value of 1 week, yet can be overridden in a contract which inherits from `MintingPointTimersManager` has been considered - it would work in our Javascript infrastructure which relies on contract source code, but not in our Python infrastructure which relies on contract byte code (since we maintain under version control the byte code of only the contracts which we are planning to deploy).

Potentially Unexpected Behavior

Duplicate Interfaces in ContractAddressLocator

In **contracts/contract_address_locator/ContractAddressLocator.sol**, the constructor doesn't verify that the received **bytes32[]** of interfaces is unique:

```
16    constructor(bytes32[] interfaceNames, address[] contractAddresses) public {
17        uint256 length = interfaceNames.length;
18        require(length == contractAddresses.length);
19        for (uint256 i = 0; i < length; i++) {
20            _registry[interfaceNames[i]] = contractAddresses[i];
21            emit Register(interfaceNames[i], contractAddresses[i]);
22        }
23    }
```

In the case that an interface name is duplicated, the last entry will override any previous entries and will result in contradicting **Register** events.

We would recommend either aborting on found duplication or documenting this as the desired behavior in both the code and the tests.

Note: we assume that duplicate addresses are the expected behavior, as a contract can implement multiple interfaces.

UPDATE: The team has resolved the issue:

Fixed to ensure that only a set of unique interface-names can be used.

Overriding Set Timestamps in MintingPointTimersManager

In **contracts/saga/MintingPointTimersManager.sol:**

```
14    function start(uint256 id) external only("IIntervalIterator") {
15        timestamps[id] = timestamp();
16    }
17
18    function reset(uint256 id) external only("IIntervalIterator") {
19        timestamps[id] = 0;
20    }
```

It's possible to set a timestamp multiple times, using the **start** method, without resetting it first, using the **reset** method.

We recommend making sure that calling **start**, for the second time, will first require calling **reset**. Besides, it'd be an excellent practice to emit respective events as the result of timestamp setting/resetting operations.

UPDATE: The team has resolved the issue:

Fixed. Moreover, I have extended the `timestamps` array from an array of integers to an array of structures. Each structure contains a timestamp and a boolean flag indicating whether or not it is valid. This way, we don't need to rely on `timestamp == 0` in order to determine that a timestamp is invalid. It helps in making the code more readable and less prone to errors.

Saga Foundation Vesting Points

According to the Saga Monetary Model, 30% of all SGN tokens, which was supposed to belong to the Saga Foundation, are subject to a vesting schedule and can't be converted to SGA, accordingly:

Vesting Point Number	1	2	3
Reserve Value at Vesting Point (SDR)	600 M	1.25 B	2 B
Market Cap at Vesting Point (SDR)	1.04 B	2.61 B	4.9 B
Percentage of total SGN unlocked at Vesting Point	10%	10%	10%

We couldn't find an explicit implementation of this behavior. We would recommend, to stay as much transparent and auditable as possible and implement this vesting behavior explicitly or to provide the right documentation and tools to verify that the distribution of SGN conforms to the vesting schedule.

UPDATE: The team has explicitly added minting points to the code.

SGN to SGA Conversion After the Last Minting Point

According to **sgn2sga** in **contracts/saga-genesis/SGNConversionManager.sol**:

```
123     function sgn2sga(uint256 amount, uint256 index) external view returns (uint256) {
124         return amount * numerators[index] / DENOMINATOR;
125     }
```

We can see that querying for the SGN to SGA conversion ratio, after reaching the final minting point, will fail with an **invalid opcode** (due to an attempt to access an out of bounds array index).

Since SGN to SGA conversion should be possible, after reaching the last minting point, we would recommend handling this case as well.

UPDATE: The team has resolved the issue.

SGN to SGA Conversion Ratio and Minting Point Discrepancy

There seems to be a discrepancy between the SGN to SGA conversion ratio, at a specific minting point, between the code and the Saga Monetary Model paper.

In "Appendix D: Genesis Minting Point", we can see that the model supports 96 genesis minting points, while in **contracts/saga-genesis/SGNConversionManager.sol** 104 genesis minting point appears, and the conversion ratios don't seem to fit:

#	ConversionManager.sol	Monetary Model Paper
0	0	0.002
1	0	0.02
2	0	0.06
3	0.02	0.13
4	0.02	0.21
5	0.06	0.32
6	0.13	0.45
7	0.13	0.6
8	0.21	0.78
9	0.32	0.98
10	0.45	1.21
11	0.6	1.21
12	0.78	1.27
13	0.98	1.32
14	0.98	1.39
15	1.21	1.45
16	1.21	1.52
17	1.27	1.59
18	1.32	1.66
19	1.39	1.73
20	1.45	1.81
21	1.52	1.81

22	1.59	1.88
23	1.66	1.96
24	1.73	2.04
25	1.81	2.12
26	1.81	2.2
27	1.88	2.29
28	1.96	2.37
29	2.04	2.46
30	2.12	2.55
31	2.2	2.55
32	2.29	2.64
33	2.37	2.72
34	2.46	2.81
35	2.55	2.9
36	2.55	2.99
37	2.64	3.09
38	2.72	3.18
39	2.81	3.28
40	2.9	3.38
41	2.99	3.47
42	3.09	3.56
43	3.18	3.68
44	3.28	3.78
45	3.38	3.89
46	3.47	3.99
47	3.58	4.1
48	3.68	4.22
49	3.78	4.33
50	3.89	4.46
51	3.99	4.58
52	4.1	4.71
53	4.22	4.85
54	4.33	4.98
55	4.46	5.13
56	4.58	5.28
57	4.71	5.43

58	4.85	5.58
59	4.99	5.74
60	5.13	5.91
61	5.28	6.08
62	5.43	6.25
63	5.58	6.43
64	5.74	6.61
65	5.91	6.79
66	6.08	6.98
67	6.08	7.18
68	6.25	7.37
69	6.43	7.57
70	6.61	7.78
71	6.79	7.99
72	6.98	8.2
73	7.18	8.42
74	7.37	8.65
75	7.57	8.87
76	7.78	9.1
77	7.99	9.34
78	8.2	9.58
79	8.42	9.82
80	8.65	10.1
81	8.87	10.3
82	9.1	10.6
83	9.34	10.9
84	9.58	11.1
85	9.82	11.4
86	10.06	11.7
87	10.32	12.1
88	10.59	12.4
89	10.86	12.7
90	11.14	13.1
91	11.44	13.4
92	11.74	13.8
93	12.06	14.1

94	12.38	14.5
95	12.38	14.9
96	12.71	15
97	13.06	-
98	13.41	-
99	13.77	-
100	14.15	-
101	14.53	-
102	14.91	-
103	15	-
104	15	-

We recommend updating the code or the paper accordingly.

UPDATE: The team has resolved the issue.

SGN Vesting Events

We can see in **contracts/saga-genesis/SGNToken.sol:**

```
101     function mintSgnVestedInDelay(uint256 _index) external only(_IMintManager_) {
102         uint256 value = valueMintedAt[_index];
103         valueMintedAt[_index] = 0;
104         getSGNTokenManager().uponMintSgnVestedInDelay(value);
105         _mint(initialOwner, value);
106     }
```

Calling the newly added **mintSgnVestedInDelay** method, as a side effect will emit two events: **MintSgnVestedInDelayCompleted** and **Transfer**.

These events will be also emitted if **mintSgnVestedInDelay** is called with an invalid index, such as **valueMintedAt[_index]** is 0, thus can unnecessarily trigger external watchers/observers.

Even though this method can only be called by the **MintManager**, we recommend avoiding emitting these events in such cases. For example, by gracefully returning if the **value** is 0 (or even reverting).

UPDATE: The team has decided not to resolve the issue and delegating its handling off-chain:

Good point, and in addition, we've noticed that the exact same issue is also relevant for `SGA Vesting Events`, with the only difference being that in SGN it is relevant for minting points

1...15, 17...25, 27...35 and 37...104, while in SGA it is relevant for minting points 1, 4, 7, 14, 16, 26, 36, 67, 95 and 104.

There are, however, a couple of notable downsides in the suggested fix:

1. *We will need to add logic in `SGNToken` and in `SGAToken`, while our purpose has been to keep these contracts as minimized as possible (since they are not upgradeable without a fork).*
2. *"Irregular" occurrence of events - at present, we have the same set of events every time the minting-point is updated, regardless of the amount being minted.*

The main upside is a slightly-reduced gas consumption, and - as the audit-report suggests - avoid triggering external watchers/observers as a result of minting 0 tokens.

However, this "redundant" event will happen very few times in the life-time of our system (more precisely - 101 times for SGN and 10 times for SGA).

So we've decided that it is not critical for us to apply this fix and that we can leave it for the off-chain server to receive these events and decide what to do with them.

Add State Modification Events to SGNWalletsTradingLimiter

We recommend to consider adding events (and respective tests) for signaling successful/failed minimum limiter value modification to **contracts/saga-genesis/SGNWalletsTradingLimiter.sol** (similarly to **contracts/saga/ETHConverter.sol**, **contracts/saga/ReconciliationAdjuster.sol**, **contracts/wallet_trading_limiter/WalletsTradingLimiterValueConverter.sol**).

In **contracts/saga-genesis/SGNWalletsTradingLimiter.sol:**

```
86      */
87      function setSGNMinimumLimiterValue(uint256 _sequenceNum, uint256 _sgnMinimumLimiterValueN, uint256 _sgnMinimumLimite
88          require(1 <= _sgnMinimumLimiterValueN && _sgnMinimumLimiterValueN <= MAX_RESOLUTION, "SGN minimum limiter value
89          require(1 <= _sgnMinimumLimiterValueD && _sgnMinimumLimiterValueD <= MAX_RESOLUTION, "SGN minimum limiter value
90
91          if (sequenceNum < _sequenceNum) {
92              sequenceNum = _sequenceNum;
93              sgnMinimumLimiterValueN = _sgnMinimumLimiterValueN;
94              sgnMinimumLimiterValueD = _sgnMinimumLimiterValueD;
95          }
96      }
```

In **contracts/saga/ETHConverter.sol:**

```
93              loPriceD = _loPriceD;
94              getTransactionLimiter().resetTotal();
95              emit PriceSaved(_hiPriceN, _hiPriceD, _loPriceN, _loPriceD);
96          }
97          else {
98              emit PriceNotSaved(_hiPriceN, _hiPriceD, _loPriceN, _loPriceD);
99          }
100     }
```

UPDATE: The team has resolved the issue.

Add Sequence Protection and State Modifications Events to RateApprover

We recommend to consider adding operation sequence protection, events (and respective tests) for signaling successful/failed high and low rate bounds **contracts/saga/RateApprover.sol** (similarly to **contracts/saga/ETHConverter.sol**, **contracts/saga/ReconciliationAdjuster.sol**, **contracts/wallet_trading_limiter/WalletsTradingLimiterValueConverter.sol**).

In **contracts/saga/RateApprover.sol**:

```
51      function setRateBounds(uint256 _maxHighRateN, uint256 _maxHighRateD, uint256 _minLowRateN, uint256 _minLowRateD) ext
52          require(1 <= _maxHighRateN && _maxHighRateN <= MAX_RESOLUTION, "max high rate numerator is out of range");
53          require(1 <= _maxHighRateD && _maxHighRateD <= MAX_RESOLUTION, "max high rate denominator is out of range");
54          require(1 <= _minLowRateN && _minLowRateN <= MAX_RESOLUTION, "min low rate numerator is out of range");
55          require(1 <= _minLowRateD && _minLowRateD <= MAX_RESOLUTION, "min low rate denominator is out of range");
56          require(_maxHighRateN * _minLowRateD > _maxHighRateD * _minLowRateN, "max high rate is smaller than min low rate
57
58          maxHighRateN = _maxHighRateN;
59          maxHighRateD = _maxHighRateD;
60          minLowRateN = _minLowRateN;
61          minLowRateD = _minLowRateD;
62      }
```

In **contracts/saga/ReconciliationAdjuster.sol**:

```
43      function setFactor(uint256 _sequenceNum, uint256 _factorN, uint256 _factorD) external onlyOwner {
44          require(1 <= _factorN && _factorN <= MAX_RESOLUTION, "adjustment factor numerator is out of range");
45          require(1 <= _factorD && _factorD <= MAX_RESOLUTION, "adjustment factor denominator is out of range");
46
47          if (sequenceNum < _sequenceNum) {
48              sequenceNum = _sequenceNum;
49              factorN = _factorN;
50              factorD = _factorD;
51              emit FactorSaved(_factorN, _factorD);
52          }
53          else {
54              emit FactorNotSaved(_factorN, _factorD);
55          }
56      }
```

UPDATE: The team has resolved the issue.

Add Event for Partial Payment Settlement

We can see in **contracts/saga/PaymentManager.sol**, that the **PaymentSettled** event is emitted for both of the cases when the payment is either settled completely or partially:

```
124         for (uint256 i = 0; i < numOfPayments; i++) {
125             (address wallet, uint256 sdrAmount) = paymentQueue.getPayment(0);
126             uint256 ethAmount = ethConverter.toEthAmount(sdrAmount);
127             uint256 ethBalance = paymentHandler.getEthBalance();
128             if (ethAmount > ethBalance) {
129                 paymentQueue.updatePayment(ethConverter.fromEthAmount(ethAmount - ethBalance)); // will never unde
130                 paymentHandler.transferEthToSgaHolder(wallet, ethBalance);
131                 emit PaymentSettled(wallet, sdrAmount, ethBalance);
132                 break;
133             }
134             paymentQueue.removePayment();
135             paymentHandler.transferEthToSgaHolder(wallet, ethAmount);
136             emit PaymentSettled(wallet, sdrAmount, ethAmount);
137         }
```

We recommend considering emitting a different **PaymentPartiallySettled** (or any other name) to easily differentiate between the cases and not entirely relying on the settlement amount.

UPDATE: The team has resolved the issue.

Integer Overflow/Underflow

The project includes many explicit arithmetic operations, some of which can result in unsigned integer overflows and underflows.

For example, in **contracts/saga-genesis/SGNConversionManager.sol:**

```
134     function sgn2sga(uint256 _amount, uint256 _index) external view returns (uint256) {
135         assert(_amount <= MAX_AMOUNT);
136         assert(_index < numerators.length);
137         return _amount * numerators[_index] / DENOMINATOR;
138     }
```

For another example, in **contracts/saga/ModelCalculator.sol**:

```
33     function getValN(uint256 _valR, uint256 _maxN, uint256 _maxR) external pure returns (uint256) {
34         return _valR.mul(_maxN) / _maxR;
35     }
```

We recommend to avoid **explicit** arithmetic operations and prefer using [OpenZeppelin's 2.0.1 SafeMath.sol] instead.

UPDATE: The team has made a great effort to wrap existing arithmetic operations with SafeMath and have documented the cases when using explicit arithmetic operations is safe.

CEI Violations

Selling of SGA to SGAToken

The SGAToken violates the recommended Checks-Effects-Interactions pattern:

In **contracts/saga/SGAToken.sol:**

```
53    function transfer(address to, uint256 value) public returns (bool) {
54        if (to == address(this)) {
55            uint256 amount = getSGATokenManager().exchangeSgaForEth(msg.sender, value);
56            msg.sender.transfer(amount);
57            _burn(msg.sender, value);
58            return true;
59        }
60        getSGATokenManager().uponTransfer(msg.sender, to, value);
61        return super.transfer(to, value);
62    }
```

External interactions should be the last step in a contract code.

We would recommend switching the order of **msg.sender.transfer(amount)** with **_burn(msg.sender, value)**.

Please note that a similar violation exists in **contracts/saga-genesis/SGNToken.sol**:

```
48    function transfer(address to, uint256 value) public returns (bool) {
49        if (to == address(this)) {
50            uint256 amount = getSGNTokenManager().exchangeSgnForSga(msg.sender, value);
51            getSagaExchanger().transferSgaToSgnHolder(msg.sender, amount);
52            _burn(msg.sender, value);
53            return true;
54        }
55        getSGNTokenManager().uponTransfer(msg.sender, to, value);
56        return super.transfer(to, value);
57    }
```

Since it doesn't result in a true external interaction (since an ERC20 **transfer** isn't interactive), this issue of a much lower priority:

UPDATE: The team has resolved the issue:

Fixed (burning SGN before transferring SGA; burning SGA before transferring ETH).

Debt Settlement

In **contracts/saga/DebtManager.sol** (older version):

```
52      function settleDebts(uint256 maxNumOfClients) external {
53          ITransactionConverter pDTransactionConverter = getTransactionConverter();
54          IDebtHandler pDebtHandler = getDebtHandler();
55          IDebtQueue pDebtQueue = getDebtQueue();
56
57          uint256 numOfClients = pDebtQueue.getNumOfClients();
58          if (numOfClients > maxNumOfClients)
59              numOfClients = maxNumOfClients;
60
61          for (uint256 i = 0; i < numOfClients; i++) {
62              (address wallet, uint256 sdrAmount) = pDebtQueue.getClient(0);
63              uint256 ethAmount = pDTransactionConverter.toEthAmount(sdrAmount);
64              uint256 ethAmountPaid = pDebtHandler.transferEthToSgaHolder(wallet, ethAmount);
65              if (ethAmount > ethAmountPaid) {
66                  pDebtQueue.updateClient(pDTransactionConverter.toSdrAmount(ethAmount - ethAmountPaid));
67                  break;
68              }
69              pDebtQueue.removeClient();
70          }
71      }
72  }
```

We can see that **transferEthToSgaHolder** is invoked before the client is being updated. This is another violation of the Checks-Effects-Interactions pattern.

We recommend changing the order of the operations by:

1. Getting the potentially paid amount.
2. Update/remove the client accordingly.
3. Transfer the owned ETH back to the client.

UPDATE: The team has resolved the issue and fixed it in the newer **PaymentManager** in full:

Fixed (removing debt-owner before transferring ETH).

Emitting Events before Checks

Even though this isn't a security concern, we recommend emitting events only during the interaction phase (i.e., after any checks and effects).

For example, in **contracts/saga/SGATokenManager.sol:**

```
109     function uponDeposit(address sender, uint256 balance) external only("ISGAToken") returns (address, uint256) {
110         (address wallet, uint256 amount) = getReserveManager().getDepositParams(balance);
111         emit Deposit(sender, balance, amount);
112         require(wallet == sender);
113         return (wallet, amount);
114     }
```

We recommend to change the order of the operations and emit the **Deposit** event only after the **require()** check.

For another example, in the same contract:

```
122     function uponWithdraw(address sender, uint256 balance) external only("ISGAToken") returns (address, uint256) {
123         (address wallet, uint256 amount) = getReserveManager().getWithdrawParams(balance);
124         emit Withdraw(sender, balance, amount);
125         require(wallet != address(0));
126         return (wallet, amount);
127     }
```

UPDATE: The team has resolved the issue.

ERC20

Redundant “Token” Suffix in SGNToken and SGAToken Names

The ERC20 token name of the **SGNToken** receives a redundant “Token” suffix:

```
13  contract SGNToken is ERC20, ContractAddressLocatorHolder {
14      string public constant name = "Saga Genesis Token";
```

Although, one can find, in the wild, examples when this suffix is used, it’s considered redundant, since most of the Ethereum block explorer append “Token” by themselves.

For example, by analyzing [ethereum-utils (4fb575)] mainnet tokens, we can see that the suffix “Token” appears only 245 times out of 1,113.

Please note that the same issue is relevant for the **SGAToken** as well:

```
16  contract SGAToken is ERC20, ContractAddressLocatorHolder, ISagaExchanger, IMintListener, IDebtHandler {
17      string public constant name = "Saga Token";
```

UPDATE: The team has resolved the issue.

Upgradability

The team chose to allow upgradability of most of the contracts using a shared registry (**contracts/contract_address_locator/ContractAddressLocator.sol**) which can be upgraded/replaced entirely via a proxy (**contracts/contract_address_locator/ContractAddressLocatorProxy.sol**).

Without debating whether full immutability is desired, the implementation grants to the **owner** of the **ContractAddressLocatorProxy** shared contract great power and incredibly responsibility, whose compromisation can have disastrous effects on the whole system,

We would recommend the team to be transparent about their keys and privileges management scheme and submit it to a security audit, thus providing the community with the ability to audit and criticize the scheme, when needed.

UPDATE: The team has stated that they will release detailed documentation regarding the upgradeability and governance:

We are planning to publish soon a document detailing Saga's governance framework.

This document will describe the cases in which the smart contract would be updated and the limitation on such updates. In addition to that, all special privilege addresses (including, of course, the owner of `ContractAddressLocatorProxy`) are going to be multisig wallets.

We are planning to publish our signatures scheme.

Front-running

Saga Core's current implementation is open to a front-running attack. Miners or any other observers, upon seeing that someone is submitting an order to buy/sell from Saga, would squeeze their own buy/sell orders ahead of the user's. They would thus get a rate from the Saga market maker that is better than what the user gets.

We don't believe that this is a significant issue since the team is planning to upgrade the current model in the future (when Saga's economy size will be sufficient to incentivize this kind of attack). In the future, we'd recommend mitigating this attack with either a commit/reveal scheme or by implementing a minimal return concept (similar to a limit order), which ensures that if the order goes below a certain level of expectation/profitability, the user can cancel it. Please note that enforcing an upper limit on gas price can reduce the race condition between users, but won't be effective against miners.

UPDATE: The team has explained that their incentives model makes front-running attack economically unprofitable:

The interplay between the increasing price-band width and the decreasing reserve ratio (both as a function of the number of SGA in circulation) makes front-running attack not profitable for the attacker in all realizable cases (See Ch. 11 of Saga's monetary model document for more details). Thus, we believe there would be no real incentive in executing a front running attack. Note that there is no plan to upgrade the model itself to mitigate the risk for front running. We are considering to implement a commit/revel mechanism and "limit order" in the future.

Testing

Test Cases Dependency

In general, it's recommended to ensure that no test case (e.g., **it()**) is dependent or affected by state changes any other test in the suite. For example, a good rule of thumb is to design the tests so that any test case will work appropriately when test cases are invoked in a **random order.**

Described below are some violations to this practice:

Avoid after()/before() Hooks

Since **after()/before()** hooks affect all the test cases in its scope, they can cause test cases to unnecessary share/reuse the state.

For example, in **test/utils/AdminableUnitTest.js** the **Adminable** instance **hAdminable** is created only once and shared between all the test cases:

```
11    before(async function() {
12        hAdminable = await artifacts.require("Adminable").new();
13    });
```

This creates an undesired state sharing and co-dependency between test cases and can lead to undesired results and reduced test quality.

We recommend making sure that every test case is using a pristine **hAdminable** instance every time by:

1. Instantiating **hAdminable** in a **beforeEach()** hook instead.
2. Setting the **hAdminable** variable to **const.**
3. Set up any state prerequisites (e.g., an **Adminable** with few pre-existing admins) in the context of the test case itself (preferably, using the **context()** alias for readability).

UPDATE: The issue wasn't resolved yet.

Avoid Test Execution Order Assumptions

Don't assume **it()** specific order of execution as it prevents any contributor from running the suites in a randomized order nor allows single/subset test debugging (e.g., using the **only** directive).

For example, in **test/authorization/AuthorizationDataSourceUnitTest.js**, the second test case assumes the correct execution of the first test case, thus obliged to use **SEQUENCE_NUM + 1** instead of **SEQUENCE_NUM**:

```
60    describe("functionality assertion:", function() {
61        it("function upsertOne should insert the wallet", async function() {
62            let response = await hAuthorizationDataSource.upsertOne(wallet, SEQUENCE_NUM, IS_AUTHORIZED, IS_RESTRICTED, TRAD
63            await verify(response.logs[0], "WalletSaved", wallet, IS_AUTHORIZED, IS_RESTRICTED, TRADING_CLASS);
64            assert((await hAuthorizationDataSource.walletCount()).equals(1));
65        });
66        it("function upsertOne should update the wallet", async function() {
67            let response = await hAuthorizationDataSource.upsertOne(wallet, SEQUENCE_NUM + 1, IS_AUTHORIZED, IS_RESTRICTED,
68            await verify(response.logs[0], "WalletSaved", wallet, IS_AUTHORIZED, IS_RESTRICTED, TRADING_CLASS);
69            assert((await hAuthorizationDataSource.walletCount()).equals(1));
70        });
71        it("function upsertOne should not update the wallet", async function() {
72            let response = await hAuthorizationDataSource.upsertOne(wallet, SEQUENCE_NUM + 1, !IS_AUTHORIZED, !IS_RESTRICTED
73            await verify(response.logs[0], "WalletNotSaved", wallet, IS_AUTHORIZED, IS_RESTRICTED, TRADING_CLASS);
74            assert((await hAuthorizationDataSource.walletCount()).equals(1));
75        });
```

UPDATE: The issue wasn't resolved yet.

Overly Permissive Test Configuration

The developer defines a **very high** gas limit of 0x1fffffffffffff in both **truffle-config.js** as well as for **ganache-cli** instance for development/tests:

In **truffle-config.js:**

```
4         development: {
5             host:       "127.0.0.1",
6             port:       8545,
7             network_id: "*",              // Match any network id
8             gasPrice:   0x1,              // Gas price used for deploys
9             gas:        0x1fffffffffffff, // Gas limit used for deploys
10        },
```

In **scripts/run-tests.js**:

```
23        "--gasPrice=0x1",
24        "--gasLimit=0x1fffffffffffff",
25        "--account=0x0000000000000000000000000000000000000000000000000000000000000001,100000000000000000000000000000000000000",
26        "--account=0x0000000000000000000000000000000000000000000000000000000000000002,100000000000000000000000000000000000000",
27        "--account=0x0000000000000000000000000000000000000000000000000000000000000003,100000000000000000000000000000000000000",
28        "--account=0x0000000000000000000000000000000000000000000000000000000000000004,100000000000000000000000000000000000000",
29        "--account=0x0000000000000000000000000000000000000000000000000000000000000005,100000000000000000000000000000000000000",
30        "--account=0x0000000000000000000000000000000000000000000000000000000000000006,100000000000000000000000000000000000000",
31        "--account=0x0000000000000000000000000000000000000000000000000000000000000007,100000000000000000000000000000000000000",
32        "--account=0x0000000000000000000000000000000000000000000000000000000000000008,100000000000000000000000000000000000000"],
33    cp.stdout.on("data", function(data) {/*process.stdout.write(data.toString());*/});
```

We recommend to set very high gas limits only during code coverage tests and allow the regulars tests to run and benchmark under mainnet-like gas limits to test various cost assumptions (e.g., making sure that the code can properly operate under acceptably stressful situations).

UPDATE: The team has resolved the issue.

Benchmarks Mixed with the Test Suite

Some of the test suites, in addition to including functional and integration test cases, also include gas benchmarks. For example, **test/authorization/AuthorizationDataSourceUnitTest.js:**

```
118     describe("performance measurement:", function() {
119         it("function upsertOne minimum gas cost per wallet", async function() {
120             let gas = await estimateGas(hAuthorizationDataSource.upsertOne, [minAddress, 1, false, false, 0]);
121             console.log(`${gas} units`);
122         });
123         it("function removeOne minimum gas cost per wallet", async function() {
124             let gas = await estimateGas(hAuthorizationDataSource.removeOne, [minAddress]);
125             console.log(`${gas} units`);
126         });
```

Since these test cases don't implement assertions or exceptions, we'd recommend excluding them from the test suite and make them optional (for example, using Mocha Tagging).

UPDATE: The team has resolved the issue.

Missing Event Tests

The following events aren't being tested in the test suite:

- In **contracts/utils/Adminable.sol**:
  - AdminAccepted
  - AdminRejected
- In **contracts/saga/SGATokenManager.sol:**
  - DepositCompleted
  - ExchangeEthForSgaCompleted
  - ExchangeSgaForEthCompleted
  - MintSgaForSgnHoldersCompleted
  - TransferEthToSgaHolderCompleted
  - TransferSgaToSgnHolderCompleted
  - WithdrawCompleted
- In **contracts/saga/MonetaryModel.sol:**
  - MonetaryModelBuyCompleted
  - MonetaryModelSellCompleted
- In **contracts/saga/PaymentManager.sol:**
  - PaymentRegistered

- PaymentSettled

We recommend testing every emitted event and its parameters.

UPDATE: The team has resolved some of the missing tests and will handle the rest in the future:

Contract Name	Event Name	Status
Adminable	AdminAccepted	Fixed
Adminable	AdminRejected	Fixed
SGNTokenManager	ExchangeSgnForSgaCompleted	Fixed
SGNTokenManager	MintSgnForFoundationCompleted	Fixed
SGATokenManager	ExchangeEthForSgaCompleted	Fixed
SGATokenManager	ExchangeSgaForEthCompleted	Fixed
SGATokenManager	MintSgaForSgnHoldersCompleted	Fixed
SGATokenManager	TransferEthToSgaHolderCompleted	Fixed
SGATokenManager	TransferSgaToSgnHolderCompleted	Fixed
SGATokenManager	DepositCompleted	Fixed
SGATokenManager	WithdrawCompleted	Fixed
MonetaryModel	MonetaryModelBuyCompleted	Fixed
MonetaryModel	MonetaryModelSellCompleted	Fixed
PaymentManager	PaymentRegistered	-
PaymentManager	PaymentSettled	Fixed

Use CI

Consider integrating publicly available automatic CI to the project (e.g., Travis CI, CircleCI, and so forth), including both invocations of the test suite as well as document its code coverage (e.g., Codecov).

UPDATE: The issue wasn't resolved yet:

Not yet handled (this is a DevOps issue).

Use Const Declarations in Tests

If a test variable is never reassigned, we would recommend declaring it as **const**. This approach prevents unintentional reassignment of constants or immutable contract instances.

For example, in **test/authorization/AuthorizationDataSourceUnitTest.js:**

```
let SEQUENCE_NUM  = 1000;
let IS_AUTHORIZED = true;
let IS_RESTRICTED = true;
let TRADING_CLASS = 2000;
```

For another example, in **test/authorization/AuthorizationDataSourceUnitTest.js:**

```
let catchRevert = require("../exceptions.js").catchRevert;
```

UPDATE: The team has resolved the issue.

Shared Constants

In some places, we can notice that important constants are being duplicated.

For example, in **test/authorization/AuthorizationDataSourceUnitTest.js:**

```
it("function upsertOne should abort with an error if the input wallet is invalid", async function() {
    await catchRevert(hAuthorizationDataSource.upsertOne("0x0", SEQUENCE_NUM, IS_AUTHORIZED, IS_RESTRICTED, TRAD
});
it("function removeOne should abort with an error if the input wallet is invalid", async function() {
    await catchRevert(hAuthorizationDataSource.removeOne("0x0", {from: admin}));
});
it("function upsertAll should abort with an error if an input wallet is invalid", async function() {
    await catchRevert(hAuthorizationDataSource.upsertAll(["0x0"], SEQUENCE_NUM, IS_AUTHORIZED, IS_RESTRICTED, TR
});
it("function removeAll should abort with an error if an input wallet is invalid", async function() {
    await catchRevert(hAuthorizationDataSource.removeAll(["0x0"], {from: admin}));
});
```

We recommend that such important constants would be shared via a library, thus allowing the compiler to detect accidental typos.

UPDATE: The team has resolved the issue.

Double Check Expectations and Configurations

We highly recommend specifying all expectations and configurations explicitly in the tests, instead of relying entirely on the Solidity implementation.

For example, in **test/saga/ModelCalculatorExtUnitTest.js:**

```
before(async function() {
    modelCalculator = await artifacts.require("ModelCalculator").new();
    A_B_SCALE = await modelCalculator.A_B_SCALE();
```

We can see that **A_B_SCALE** is being retrieved from the contract and later used directly, but what will happen if the **A_B_SCALE** was defined incorrectly in the contract? The tests will pass since they all build upon the incorrect value.

We recommend defining it explicitly in the test as well and comparing it to the retrieved value from the contract. This is also relevant for other configuration values and assumptions.

UPDATE: The team has resolved this issue in most of the places..

Time Sensitive Tests

We recommend verifying time-sensitive/aware tests using time manipulation capabilities, provided by **ganache** (for example, using [OpenZeppelin's time.js helper]), instead of state manipulating mockups.

For example, in **test/saga/MintManagerUnitTest.js:**

```
16      describe("functionality assertion:", function() {
17          before(async function() {
18              await hContractAddressLocatorProxy.set("IDataSource"  , hDataSource  .address);
19              await hContractAddressLocatorProxy.set("ITimeManager" , hTimeManager .address);
20              await hContractAddressLocatorProxy.set("IMintListener", hMintListener.address);
21          });
22          for (let expired of [false, true]) {
23              for (let index = 0; index < 10; index++) {
24                  it(`expired = ${expired}, index = ${index}`, async function() {
25                      await hTimeManager.setExpired(expired);
26                      await hMintManager.setIndex(index);
27                      let state = await hMintManager.isMintingStateOutdated();
28                      await hMintManager.updateMintingState();
29                      let expected = state ? index + 1 : index;
30                      let actual = await hMintManager.getIndex();
31                      assert(actual.equals(expected), `expected = ${expected}, actual = ${actual}`);
32                  });
33              }
34          }
35      });
36  });
```

Instead of allowing for the interval to expire naturally (e.g., by manipulating the time using **time.js**'s **increase** or **increaseTo**) methods, the test forces the internal to expire using the **TimeManagerMock**up in **contracts/saga/helpers/MintingPointTimersManagerMockup.sol:**

```
31      function setExpired(bool _state) external {
32          isExpired = _state;
33      }
```

Besides, the **setExpired** method expires **all** the intervals (similarly to **setRunning** setting **all** the intervals to running), thus dangerously affecting the state globally.

UPDATE: The team has reassured that time manipulation via mockups is only used in unrelated unit-tests:

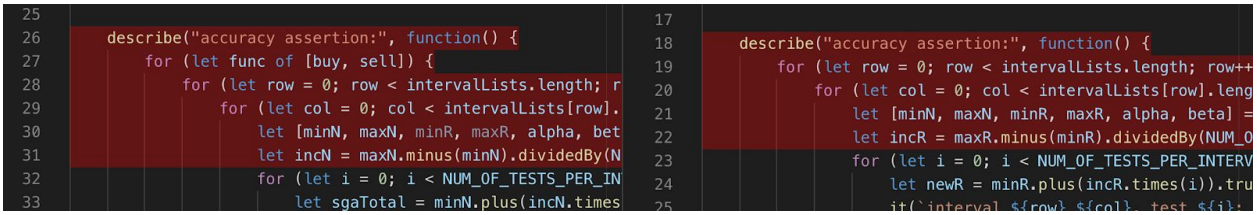
The only contract which depends on time is `TimeManager`. And indeed, in this contract's unit-test, we manipulate the time via `evm_increaseTime`.

In all other unit-tests, since we use the `TimeManagerMockup` contract instead, we have no need to manipulate the time in this manner.

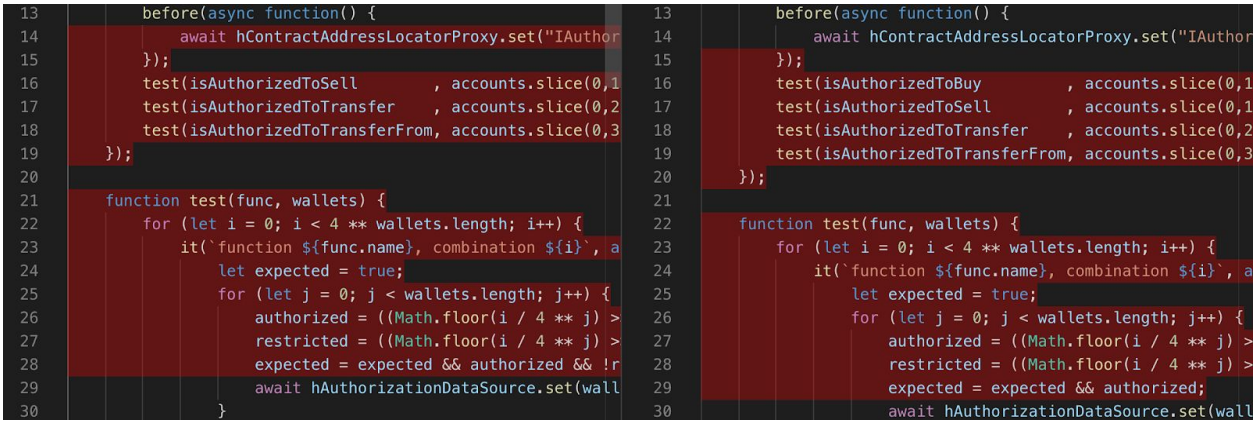
Code Duplication

There appears to be some code duplication and repetition in tests.

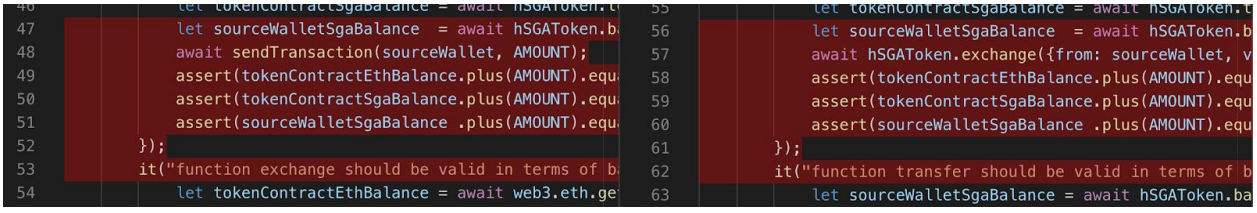
For example, between **test/saga/PriceBandCalculatorExtUnitTest.js** and **test/saga/ModelCalculatorExtUnitTest.js:**



For another example, between **test/saga-genesis/SGNAuthorizationManagerUnitTest.js** and **test/saga/SGAAuthorizationManagerUnitTest.js:**



For another example, in **test/saga/SGATokenUnitTest.js:**



We recommend reusing shared logic in tests (using a shared function, utilizing Mocha Shared Behaviours, and so forth).

UPDATE: The team hasn't resolved these issues yet since it's a low priority.

Semantic Assertions

We recommend asserting equality using the **equals** instead of directly comparing the objects and asserting the truthfulness of the comparison. In doing so, an assertion error will be much more descriptive and explicitly show the differences between the **expected** and the **actual** results.

For example, in **test/authorization/AuthorizationDataSourceUnitTest.js:**

```
156
157     async function verify(log, event, wallet, isAuthorized, isRestricted, tradingClass) {
158         assert(log.event == event);
159         assert(log.args.wallet == wallet);
160         assert((await hAuthorizationDataSource.isAuthorized(wallet)) == isAuthorized);
161         assert((await hAuthorizationDataSource.isRestricted(wallet)) == isRestricted);
```

Can be refactored as:

```
157     async function verify(log, event, wallet, isAuthorized, isRestricted, tradingClass) {
158         assert(log.event).equals(event);
159         assert(log.args.wallet).equals(wallet);
160         assert((await hAuthorizationDataSource.isAuthorized(wallet))).equals(isAuthorized);
161         assert((await hAuthorizationDataSource.isRestricted(wallet))).equals(isRestricted);
```

If you do choose to use boolean **assert**s, then please consider specifying a message, in case the assertions fail.

UPDATE: The team has resolved the issue.

Require Node Modules at the Beginning of the File

We recommend avoiding requiring node modules in the middle of the test files, to prevent any module loading side effects during the tests. It is also better to use it at the beginning of your file to avoid blocking calls while your tests run.

For example, in **test/saga/PaymentManagerUnitTest.js:**

```
44     let catchRevert = require("../exceptions.js").catchRevert;
```

UPDATE: The team hasn't resolved the issue yet.

Documentation

README.md

We would recommend enhancing the documentation by adding to README.md::

1. Document project's prerequisites in **README.md.**
2. Add testing and code coverage running instructions to **README.md.**
3. Consider either referencing or embedding **docs/** in **README.md.**

UPDATE: The team has resolved the issue.

SECURITY.md

We highly recommend adding SECURITY.md file to every repository and standardize the way security incidents are **responsibly** disclosed.

SECURITY.md should describe:

1. How to report security incidents and vulnerabilities.
2. How to report anonymously (e.g., not to be blamed if the vulnerability is being exploited).
3. How to make sure that the report is being delivered to the right personnel (e.g., by providing an email as well as a PGP key.
4. Information regarding any bounty programs and rewards.

Please see Appendix F: SECURITY.md Template for an example.

We also recommend adding your SECURITY.md using Github's new Security Policy feature. Please see Appendix G: Setting Github Security Policy for more information.

Maintainer Security Advisory

I highly recommend using Github's new Maintainer Security Advisory feature (which is currently in beta but works like a charm nonetheless). Now maintainers can have a private workspace to discuss, fix, and publish security advisories to people who rely on their projects right within GitHub (without tipping off would-be hackers).

Please see Appendix H: Setting up Maintainer Security Advisory for more information.

Mechanism

SGA_MINTED_FOR_SGN_HOLDERS

Document and emphasize the purpose of the **SGA_MINTED_FOR_SGN_HOLDERS** (0x10063FCCf5eEE46fC65D399a7F5dd88730906CF9) address. Without proactive explanation, the community might confuse this reservation mechanism with unfair SGN token allocation.

UPDATE: The team has resolved the issue.

PriceBandCalculator Parameters

Document how the parameters in **contracts/saga/PriceBandCalculator.sol** were selected:

```
9     uint256 public constant ONE    = 1000000000;
10    uint256 public constant GAMMA  = 179437500000000000000000000000000000000;
11    uint256 public constant DELTA  = 29437500;
12    uint256 public constant BUY_N  = 2000;
13    uint256 public constant BUY_D  = 2003;
14    uint256 public constant SELL_N = 1997;
15    uint256 public constant SELL_D = 2000;
```

UPDATE: The team has resolved the issue.

ModelCalculator Precision Parameters

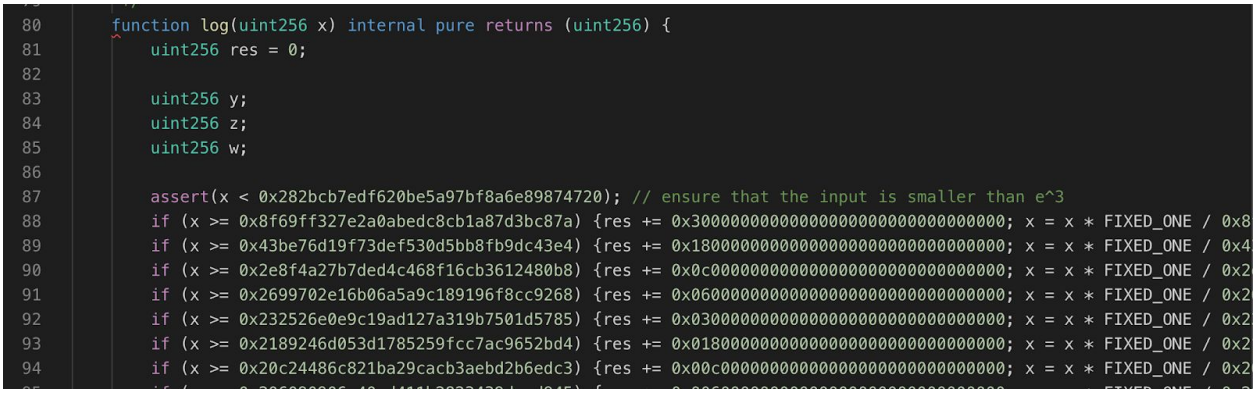
Document how the precision parameters in **contracts/saga/ModelCalculator.sol** were selected:

```
6     uint256 public constant FIXED_ONE = 0x200000000000000000000000000000000;
7     uint256 public constant A_B_SCALE = 100000000000000000000000000000000;
```

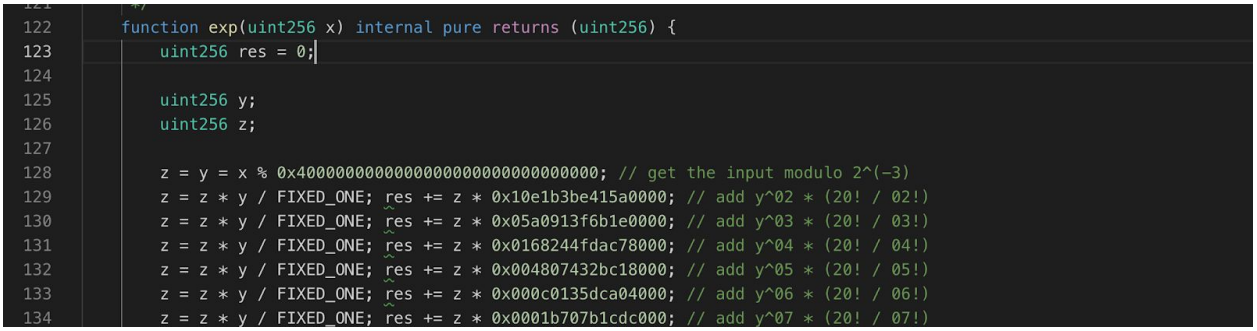
UPDATE: The team has resolved the issue.

Natural Logarithm and Exponential Functions in PriceCalculator

Document how the precision and the numerical estimation method were chosen in **contracts/saga/ModelCalculator.sol:**



```
80  function log(uint256 x) internal pure returns (uint256) {
81      uint256 res = 0;
82
83      uint256 y;
84      uint256 z;
85      uint256 w;
86
87      assert(x < 0x282bcb7edf620be5a97bf8a6e89874720); // ensure that the input is smaller than e^3
88      if (x >= 0x8f69ff327e2a0abedc8cb1a87d3bc87a) {res += 0x3000000000000000000000000000000; x = x * FIXED_ONE / 0x8
89      if (x >= 0x43be76d19f73def530d5bb8fb9dc43e4) {res += 0x1800000000000000000000000000000; x = x * FIXED_ONE / 0x4
90      if (x >= 0x2e8f4a27b7ded4c468f16cb3612480b8) {res += 0x0c0000000000000000000000000000; x = x * FIXED_ONE / 0x2
91      if (x >= 0x2699702e16b06a5a9c189196f8cc9268) {res += 0x0600000000000000000000000000000; x = x * FIXED_ONE / 0x2
92      if (x >= 0x232526e0e9c19ad127a319b7501d5785) {res += 0x0300000000000000000000000000000; x = x * FIXED_ONE / 0x2
93      if (x >= 0x2189246d053d1785259fcc7ac9652bd4) {res += 0x0180000000000000000000000000000; x = x * FIXED_ONE / 0x2
94      if (x >= 0x20c24486c821ba29cacb3aebd2b6edc3) {res += 0x00c0000000000000000000000000000; x = x * FIXED_ONE / 0x2
```



```
122 function exp(uint256 x) internal pure returns (uint256) {
123     uint256 res = 0;
124
125     uint256 y;
126     uint256 z;
127
128     z = y = x % 0x4000000000000000000000000000000; // get the input modulo 2^(-3)
129     z = z * y / FIXED_ONE; res += z * 0x10e1b3be415a0000; // add y^02 * (20! / 02!)
130     z = z * y / FIXED_ONE; res += z * 0x05a0913f6b1e0000; // add y^03 * (20! / 03!)
131     z = z * y / FIXED_ONE; res += z * 0x0168244fdac78000; // add y^04 * (20! / 04!)
132     z = z * y / FIXED_ONE; res += z * 0x004807432bc18000; // add y^05 * (20! / 05!)
133     z = z * y / FIXED_ONE; res += z * 0x000c0135dca04000; // add y^06 * (20! / 06!)
134     z = z * y / FIXED_ONE; res += z * 0x0001b707b1cdc000; // add y^07 * (20! / 07!)
```

Even though the estimation method is already documented in the code, we highly recommend providing extensive and highly detailed additional documentation is essential for increasing the project's auditability and transparency.

UPDATE: The team has resolved the issue.

Reasonable Gas Expectations in MonetaryModel

We can see in **contracts/saga/MonetaryModel.sol** that both **buyFunc** and **sellFunc** has potentially gas exhaustive while loops:

```
100     function buyFunc(uint256 _sdrAmount, ISagaModelState _sagaModelState, IIntervalIterator _intervalIterator) private r
101         uint256 sgaCount = 0;
102         uint256 sdrCount = _sdrAmount;
103
104         uint256 sdrDelta;
105         uint256 sgaDelta;
106
107         uint256 sdrTotal = _sagaModelState.getSdrTotal();
108         uint256 sgaTotal = _sagaModelState.getSgaTotal();
109
110         (uint256 minN, uint256 maxN, uint256 minR, uint256 maxR, uint256 alpha, uint256 beta) = _intervalIterator.getCur
111         while (sdrCount >= maxR.sub(sdrTotal)) {
112             sdrDelta = maxR.sub(sdrTotal);
113             sgaDelta = maxN.sub(sgaTotal);
114             _intervalIterator.grow();
115             (minN, maxN, minR, maxR, alpha, beta) = _intervalIterator.getCurrentInterval();
116             sdrTotal = minR;
117             sgaTotal = minN;
118             sdrCount = sdrCount.sub(sdrDelta);
119             sgaCount = sgaCount.add(sgaDelta);
120         }
```

```
156         (uint256 minN, uint256 maxN, uint256 minR, uint256 maxR, uint256 alpha, uint256 beta) = _intervalIterator.getCur
157         while (sgaCount > sgaTotal.sub(minN)) {
158             sgaDelta = sgaTotal.sub(minN);
159             sdrDelta = sdrTotal.sub(minR);
160             _intervalIterator.shrink();
161             (minN, maxN, minR, maxR, alpha, beta) = _intervalIterator.getCurrentInterval();
162             sgaTotal = maxN;
163             sdrTotal = maxR;
164             sgaCount = sgaCount.sub(sgaDelta);
165             sdrCount = sdrCount.add(sdrDelta);
166         }
```

We recommend documenting why the current model parameters will make the convergence of these loops in reasonable gas constraints.

UPDATE: The team has resolved the issue.

Code

Missing Methods/Constructors Documentation

There are many methods/constructors with missing parameter and expectations documentation.

For example, in **contracts/contract_address_locator/ContractAddressLocator.sol:**

```
16      constructor(bytes32[] interfaceNames, address[] contractAddresses) public {
17          uint256 length = interfaceNames.length;
18          require(length == contractAddresses.length);
19          for (uint256 i = 0; i < length; i++) {
20              _registry[interfaceNames[i]] = contractAddresses[i];
21              emit Register(interfaceNames[i], contractAddresses[i]);
22          }
23      }
```

We highly recommend making sure that every constructor, as well as at least **public/external** methods, are documented.

UPDATE: The team has resolved the issue.

Missing Interfaces Documentation

We highly recommend documenting the interfaces explicitly. These are most likely will be solely exposed to external collaborators, thus documenting them is as vital as documenting their concrete implementations.

UPDATE: The team has resolved the issue.

Missing Formulas and Calculation

We recommend that the following formulas and calculation would be explicitly documented (and possibly include references to the model):

- Conversion in **numerators** in **contracts/saga-genesis/SGNConversionManager.sol:**

```
20    constructor() public {
21        numerators[  0] =        0;
22        numerators[  1] =        0;
23        numerators[  2] =     2500;
24        numerators[  3] =    22650;
25        numerators[  4] =    22650;
26        numerators[  5] =    63504;
27        numerators[  6] =   125680;
28        numerators[  7] =   125680;
29        numerators[  8] =   209810;
```

UPDATE: The team has resolved the issue.

Add Versions to Contracts

Since many of the contracts are upgradable, we highly recommend to include a version string in each upgradable contract. For example, by adding a **VERSION** public constant, such as:

```
13    string public constant VERSION = "1.0.0";
```

UPDATE: The team has resolved the issue.

Miscellaneous

External Package Modification

In **package.json** we can see that the developer specifies an additional install script:

```
6   "scripts": {
7     "install": "node scripts/fix-modules.js",
8     "build": "node scripts/rebuild-all.js",
```

In addition to (relatively) simple patching of **truffle** and **solidity-coverage**, **fix-modules.js** is responsible for replacing **openzeppelin-solidity** 2.0.0's **Ownable.sol** and imports **Claimable.sol** smart contracts to their 1.12.0 counterparts **in place**:

```
29  for (let filename of ["Ownable.sol", "Claimable.sol"]) {
30      let url = "https://raw.githubusercontent.com/OpenZeppelin/openzeppelin-solidity/v1.12.0/contracts/ownership/" + filename;
31      let options = {directory: "./node_modules/openzeppelin-solidity/contracts/ownership/", filename: filename};
32      console.log("Fixing " + options.directory + options.filename);
33      df(url, options, function(error) {if (error) throw error;});
34  }
35
```

This is a dangerous practice as it modifies **openzeppelin-solidity** in a potentially unexpected way.

We recommend to avoid any such modifications and elect on working with the official, pristine, and audited version of **openzeppelin-solidity**. If an additional behavior is required (such as **Claimable.sol**), it should be added and tested explicitly with the rest of the developer provided contracts.

UPDATE: The team has resolved the issue:

Fixed by downloading `Ownable` and `Claimable` into `openzeppelin-solidity-v1.12.0` (instead of `openzeppelin-solidity`). This way, the following is achieved:

1. *Package installed by `npm` (openzeppelin-solidity v2.0.0) is not modified.*
2. *Anyone reading the contracts is fully aware of the fact that `Ownable` and `Claimable` belong to `openzeppelin-solidity-v1.12.0` (because this string appears explicitly in the `import` statements).*
3. *Saga leverages from the fact that these modules are well-known open-source, and have been verified thoroughly by the community (which would not be the case should we choose to implement them ourselves).*

Outdated Dependencies

The project references the following outdated/obsolete dependencies which should be considered to be updated:

1. **truffle** 4.1.14: please consider updating to 5.0.30. Please note that the latest versions support specifying custom Solidity version.
2. **ganache-cli** 6.4.1: please consider updating to 6.5.1.
3. **solidity-coverage:** please consider updating to 0.6.4.
4. **truffle-flattener:** please consider updating to 1.4.0.
5. **web3** 1.0.0-beta.34: updating to **truffle** 5.0.0 will also allow upgrading **web3** to the official release 1.2.0.

We highly recommend upgrading to Solidity to a bug fix version 0.4.26.

Error Handling

More Descriptive require()/revert()

It's recommended to provide a message string for either **require()** or **revert()** to provide more descriptive errors messages in tests (and in the future, also in mainnet).

For example, the **require()** in **utils/Adminable.sol:**

```
30    function accept(address admin) external onlyOwner {
31        require(admin != address(0));
```

Can be replaced with:

```
30    function accept(address admin) external onlyOwner {
31        require(admin != address(0), "Adminable::accept - admin address cannot be zero!");
```

Which produces the following error message during tests:

```
Error: VM Exception while processing transaction: revert Adminable::accept - admin address can't be 0x0!
```

For another example, in **contracts/saga-genesis/SGNToken.sol**:

```
67    function transferFrom(address from, address to, uint256 value) public returns (bool) {
68        if (to == address(this)) {
69            revert();
70        }
```

For another example, in **contracts/saga/SGAToken.sol:**

```
72    function transferFrom(address from, address to, uint256 value) public returns (bool) {
73        if (to == address(this)) {
74            revert();
75        }
```

UPDATE: The team has resolved the issue.

Use Assertions for Verifying Conditions Which Should Never Be Possible

Prefer **assert()** over **require()** when verifying the correctness of internal state and logic to prevent anything terrible from happening, while **require()** should be used to ensure valid conditions, such as inputs, or contract state variables are met, or to validate return values from calls to external contracts.

For example, in **contracts/saga-genesis/SGNTokenManager.sol:**

```
40    function exchangeSgnForSga(address sender, uint256 sgnAmount) external only("ISGNToken") returns (uint256) {
41        require(getSGNAuthorizationManager().isAuthorizedToSell(sender));
42        uint256 sgaAmount = _convertSgnToSga(sgnAmount);
43        require(sgaAmount > 0);
44        return sgaAmount;
45    }
```

The **require(sgaAmount > 0)** guard should be considered to be replaced with an **assert()**.

By following this guidance, static analysis and formal verification tools will be able to examine the contracts to and prove that they operate as designed without flaws.

UPDATE: The team has resolved the issue.

Overly Aggressive Optimization

The developer sets **solc** optimizer runs to 5,000,000.

In **truffle-config.js**:

```
34    solc: {
35        optimizer: {
36            enabled: true,
37            runs:    5000000,
38        },
39    },
```

Even though the community has started to experiment with such aggressive optimizations settings (e.g., as suggested here), it's relatively new and not as battle-tested as the default 200 optimizer runs.

Our recommendation is to avoid such an aggressive optimization, **for the time being,** and prefer the default settings instead. If the aggressive optimization is required to support expected scenarios, we'd recommend reverting to semantic and other low-hanging fruit optimizations.

UPDATE: The team has elected to reduce opimizer-runs to 6000:

The `optimizer-runs` parameter affects the trade-off between deployment cost and runtime cost. From the official documentation (of solc v0.5.3, due to a recent clarification). As stated by chriseth (solc chief engineer) in this GitHub thread, the `optimizer-runs` parameter

determines only how constants and literals are encoded, and we should be safe as long as we have tests that access every constant once and call every function once.

We have therefore configured it to 6000 - the minimum value which yields maximum optimization.

Constants

Use Units and Denominations

We recommend using more descriptive units and denominations when defining constants.

For example, in **contracts/saga-genesis/SGNToken.sol**, the **TOTAL_SUPPLY** constant:

```
18    uint256 public constant TOTAL_SUPPLY = 10700000000000000000000000;
```

Can be defined as:

```
18    uint256 public constant TOTAL_SUPPLY = 107000000 * uint(10) ** decimals;
```

Alternatively, using another constant for the denomination:

```
18    uint256 public constant TOKEN_UNIT = 10 ** 18;
19    uint256 public constant TOTAL_SUPPLY = 107000000 * TOKEN_UNIT;
```

UPDATE: The team has resolved the issue.

Use Postfix Exponents

We recommend, wherever applicable, to define the powers of 10 constants using the **e** postfix exponent.

For example, in **contracts/saga-genesis/SGNConversionManager.sol**, the **DENOMINATOR** constant:

```
7     uint256 public constant DENOMINATOR = 1000000;
```

Can be defined as:

```
7     uint256 public constant DENOMINATOR = 1e6;
```

UPDATE: The team has decided to keep the current style:

We prefer to use 1000000 (and not 1e6 as you suggested) because we'd like to display an easily readable set of conversion ratios (105 values between 0 and 15).

Since the `numerators` are all denoted in full format, we believe that we may as well denote the `DENOMINATOR` in the same manner.

Shared Constants

In some places, we can notice that important constants are being duplicated.

For example, in **contracts/saga/SGAToken.sol:**

```
25    function getSGATokenManager() public view returns (ISGATokenManager) {
26        return ISGATokenManager(get("ISGATokenManager"));
27    }
```

In **contracts/saga/TransactionManager.sol:**

```
32    function buy(uint256 ethAmount) external only("ISGATokenManager") returns (uint256) {
33        uint256 sdrAmount = getTransactionConverter().toSdrAmount(ethAmount);
34        uint256 newAmount = getInterestConverter().adjustBuy(sdrAmount);
35        uint256 sgaAmount = getSagaModel().buy(newAmount);
36        getTransactionLimiter().incTotalBuy(sdrAmount);
37        emit TransactionManagerBuy(sdrAmount);
38        return sgaAmount;
39    }
40
41    function sell(uint256 sgaAmount) external only("ISGATokenManager") returns (uint256) {
42        uint256 sdrAmount = getSagaModel().sell(sgaAmount);
43        uint256 newAmount = getInterestConverter().adjustSell(sdrAmount);
```

We recommend that such important constants (especially Registry related) would be shared via a library, thus allowing the compiler to detect accidental typos.

For example, having a **Constants.sol**:

```
1  pragma solidity 0.4.24;
2
3
4  library Constants {
5      function ISGA_TOKEN_MANAGER() internal pure returns (bytes32) {
6          return "ISGATokenManager";
7      }
8  }
9
```

Which can be used in turn as:

```
25
26      function getSGATokenManager() public view returns (ISGATokenManager) {
27          return ISGATokenManager(get(Constants.ISGA_TOKEN_MANAGER()));
28      }
```

UPDATE: The team has resolved the issue via **contracts/contract_address_locator/ContractAddressLocatorHolder.sol**.

Token Amounts

We recommend defining constants, defining full token amounts, using token decimals.

For example, in **contracts/saga-genesis/SGNConversionManager.sol:**

```
14    uint256 public constant MAX_AMOUNT = 107e24;
```

MAX_AMOUNT implicitly assumes 10^{18} precision of the SGN token.

We recommend to define it using **SGNToken** decimals explicitly (e.g., by passing an instance of the **SGNToken**) or explicitly documenting it.

UPDATE: The team has resolved the issue.

State Variables

Prefer Initializing State Variables in their Declarations

We recommend initializing state variables in their declarations (instead of in the constructor) to avoid their initialization during constructor rewrites or refactoring.

For example, the initialization in **contracts/saga/ModelDataSource.sol:**

```
16    bool public intervalListsUninitialized;
17    Interval[11][105] public intervalLists;
18
19    /**
20     * @dev Create the contract
21     */
22    constructor() public {
23        intervalListsUninitialized = true;
24    }
```

For another example, in **contracts/saga/ReconciliationAdjuster.sol:**

```
12    uint256 public sequenceNum;
13    uint256 public sdrFactorN;
14    uint256 public sdrFactorD;
15
16    constructor() public {
17        sequenceNum = 0;
18        sdrFactorN = 1;
19        sdrFactorD = 1;
20    }
```

UPDATE: The team has resolved the issue.

Redundant Getters

In some places, for public state variables, an additional explicit getter is defined.

For example, in **contracts/saga/MintManager.sol:**

```
9  contract MintManager is IMintManager, ContractAddressLocatorHolder {
10     uint256 public index;
```

We can later see that an explicit **getIndex** getter is defined:

```
38     function getIndex() external view returns (uint256) {
39         return index;
40     }
```

For another example, in **contracts/saga/MonetaryModelState.sol**:

```
20     function getSdrTotal() external view returns (uint256) {
21         return sdrTotal;
22     }
23
24     function getSgaTotal() external view returns (uint256) {
25         return sgaTotal;
26     }
```

Since that, for public state variables, an automatic getter function is generated, we recommend to avoid defining additional explicit getters.

Please note, that in some cases, due to limitations of the web.js framework, this is unavoidable.

UPDATE: The team has decided to keep these additional getters for improved code readability and consistency:

These getters are not redundant. Because we use them also from the on-chain (and not only from the off-chain):

1. *We want to use names which start with `get` and naming a variable this way would be very misleading.*
2. *Solidity's implicit getters cannot be properly exposed in an `interface`, due to a technical limitation (the compiler generates `public` getters, while `interface` functions must be declared `external`).*

Explicit Types

We recommend to define state variable types and avoid using **uint** explicitly. Although **uint** is an alias for **uint256**, it makes the size implicit, for the only benefit of having a few characters less.

For example, in **contracts/authorization/AuthorizationDataSource.sol:**

```
6   contract AuthorizationDataSource is IAuthorizationDataSource, Adminable {
7       uint public walletCount;
8
9       struct WalletInfo {
10          uint sequenceNum;
11          bool isAuthorized;
12          bool isRestricted;
13          uint tradingClass;
14      }
```

UPDATE: The team has resolved the issue.

Registry Getters

In most of the cases, a contract defined a getter for a specific interface from the registry. For example, in **contracts/saga/TradingManager.sol**:

```
13      function getAuthorizationDataSource() public view returns (IAuthorizationDataSource) {
14          return IAuthorizationDataSource(get("IAuthorizationDataSource"));
15      }
16
17      function getTradingDataSource() public view returns (ITradingDataSource) {
18          return ITradingDataSource(get("ITradingDataSource"));
19      }
20
21      function getTradingConverter() public view returns (ITradingConverter) {
22          return ITradingConverter(get("ITradingConverter"));
23      }
```

Yet, in **contracts/saga/TransactionConverter.sol** the registry is accessed directly:

```
41              ITransactionLimiter(get("ITransactionLimiter")).resetTotal();
```

We recommend changing the above to stay consistent with the rest of the code.

UPDATE: The team has resolved the issue.

Local Variables

Object Notation for Struct Initialization

We recommend initializing structs using object notation, for improved readability and accidental order switching of members with the same type.

For example, the initialization in **contracts/saga/ModelDataSource.sol:**

```
44      function setInterval(uint256 rowNum, uint256 colNum, uint256 minN, uint256 maxN, uint256 minR, uint256 maxR, uint256
45          require(intervalListsUninitialized);
46          intervalLists[rowNum][colNum] = Interval(minN, maxN, minR, maxR, alpha, beta);
47      }
```

Can be written as (please note that intentional ordering of **maxR**, which the proposed initialization is agnostic to):

```
43      function setInterval(uint256 rowNum, uint256 colNum, uint256 minN, uint256 maxN, uint256 minR, uint256 maxR, uint256
44          require(intervalListsUninitialized);
45          intervalLists[rowNum][colNum] = Interval({minN: minN, maxN: maxN, maxR: maxR, minR: minR, alpha: alpha, beta: be
46      }
```

UPDATE: The team has resolved the issue.

Use Memory Non-Persistent Variables

Unless a change to a state variable is required to persist, we recommend using **memory** local variables instead (**memory** variables are temporary variables that exist only inside the calling function and get wiped after the function exits, thus are generally cheaper to use than **storage** variables).

For example, in **contracts/saga/ModelDataSource.sol:**

```
60      function getInterval(uint256 rowNum, uint256 colNum) external view returns (uint256, uint256, uint256, uint256, uint
61          Interval storage interval = intervalLists[rowNum][colNum];
62          return (interval.minN, interval.maxN, interval.minR, interval.maxR, interval.alpha, interval.beta);
63      }
```

Can be refactored to:

```
60      function getInterval(uint256 rowNum, uint256 colNum) external view returns (uint256, uint256, uint256, uint256, uint
61          Interval memory interval = intervalLists[rowNum][colNum];
62          return (interval.minN, interval.maxN, interval.minR, interval.maxR, interval.alpha, interval.beta);
63      }
```

Please note that this issue appears in other places as well.

UPDATE: The team has concluded and showed that the following changes don't yield performance optimizations in their case and rightfully decided to keep their code unchanged:

I've conducted several tests, all of which have unanimously shown that declaring the local variable as `memory` yields a higher gas-consumption than declaring it as `storage`. This is also visible in the disassembly (several opcodes added when replacing `storage` with `memory`). Not fixing.

Explicit Types

We recommend to define local variable types and avoid using **uint** explicitly. Although **uint** is an alias for **uint256**, it makes the size implicit, for the only benefit of having a few characters less.

For example, in **contracts/authorization/AuthorizationDataSource.sol:**

```
43      function upsertAll(address[] wallets, uint sequenceNum, bool isAuthorizedVal, bool isRestrictedVal, uint tradingClas
44          for (uint i = 0; i < wallets.length; i++)
45              _upsert(wallets[i], sequenceNum, isAuthorizedVal, isRestrictedVal, tradingClassVal);
46      }
47
48      function removeAll(address[] wallets) external onlyAdmin {
49          for (uint i = 0; i < wallets.length; i++)
50              _remove(wallets[i]);
51      }
```

UPDATE: The team has resolved the issue.

Cache External Calls

In some cases, it's advised to cache external call results (e.g., registry entries), to reduce gas costs.

For example, in **contracts/saga/MonetaryModel.sol:**

```
50      function _buy(uint256 sdrAmount) private returns (uint256) {
51          uint256 sgaCount = 0;
52          uint256 sdrCount = sdrAmount;
53
54          uint256 sdrDelta;
55          uint256 sgaDelta;
56
57          uint256 sdrTotal = getSagaModelState().getSdrTotal();
58          uint256 sgaTotal = getSagaModelState().getSgaTotal();
59
60          (uint256 minN, uint256 maxN, uint256 minR, uint256 maxR, uint256 alpha, uint256 beta) = getIntervalIterator().ge
61          while (sdrCount >= maxR - sdrTotal) {
62              sdrDelta = maxR - sdrTotal;
63              sgaDelta = maxN - sgaTotal;
64              getIntervalIterator().grow();
65              (minN, maxN, minR, maxR, alpha, beta) = getIntervalIterator().getCurrentInterval();
66              sdrTotal  = minR;
67              sgaTotal  = minN;
68              sdrCount -= sdrDelta;
69              sgaCount += sgaDelta;
70          }
71
72          if (sdrCount > 0) {
73              if (getPriceCalculator().isTrivialInterval(alpha, beta))
74                  sgaDelta = getPriceCalculator().getValN(sdrCount, maxN, maxR);
75              else
76                  sgaDelta = getPriceCalculator().getNewN(sdrTotal + sdrCount, minN, minR, alpha, beta) - sgaTotal;
77              sdrTotal += sdrCount;
78              sgaTotal += sgaDelta;
79              sgaCount += sgaDelta;
80          }
81
82          getSagaModelState().setSdrTotal(sdrTotal);
83          getSagaModelState().setSgaTotal(sgaTotal);
84
85          return sgaCount;
86      }
```

Every external call, to the registry, could have been cached.

Please note, that In this specific case, caching all external calls will result in exceeding the allowed amount of locally defined variables (in a “Stack too deep” compilation error), which is a good hint that the function is slightly too complicated and should be split into subfunctions.

UPDATE: The team has resolved the issue:

This issue appears in the `TimeManager` contract and in the `SagaModel` contract.

In `TimeManager` - fixed.

In `SagaModel`, due to the stack-overflow problem, we can cache only 2 out of 3 addresses:

1. *`SagaModelState` contract, which is fetched 5 times on every buy/sell operation.*
2. *`IntervalIterator` contract, which is fetched 4 times on almost every buy/sell operation.*
3. *`PriceCalculator` contract, which is fetched 2 times on almost every buy/sell operation.*

In order to achieve maximum improvement, we chose to cache the first two options above. This yields an improvement of 10-15% (around 10% in most cases).

Modifiers

Use Modifiers for State Verification

We recommend to use modifiers, instead of **require()** guards, whenever the check represents state enforcement (and not, for example, input validation).

For example, in **contracts/saga/ModelDataSource.sol**:

```
44      function setInterval(uint256 rowNum, uint256 colNum, uint256 minN, uint256 maxN, uint256 minR, uint256 maxR, uint256
45          require(intervalListsUninitialized);
```

The **require** can be refactored to an **onlyUninitialized** modifier.

For another example, in **contracts/saga/SGATokenManager.sol:**

```
53      function exchangeEthForSga(address sender, uint256 ethAmount) external only("ISGAToken") returns (uint256) {
54          require(!getRedButton().isEnabled());
```

```
68      function exchangeSgaForEth(address sender, uint256 sgaAmount) external only("ISGAToken") returns (uint256) {
69          require(!getRedButton().isEnabled());
```

In both of the cases, the **require** can be refactored to an **onlyEnabled/whenEnabled** modifier.

For another example, in **contracts/saga/ModelDataSource.sol:**

```
29      function lock() external onlyOwner {
30          intervalListsUninitialized = false;
31      }
32
33      /**
34       * @dev   Set interval parameters
35       * @param rowNum - Interval row index
36       * @param colNum - Interval column index
37       * @param minN   - Interval minimum number of SGA
38       * @param maxN   - Interval maximum number of SGA
39       * @param minR   - Interval minimum number of SDR
40       * @param maxR   - Interval maximum number of SDR
41       * @param alpha  - Interval alpha value (scaled up)
42       * @param beta   - Interval beta  value (scaled up)
43       */
44      function setInterval(uint256 rowNum, uint256 colNum, uint256 minN, uint256 maxN, uint256 minR, uint256 maxR, uint256
45          require(intervalListsUninitialized);
46          intervalLists[rowNum][colNum] = Interval(minN, maxN, minR, maxR, alpha, beta);
47      }
```

UPDATE: The team has resolved the issue:

I have conducted research on the considerations for when to use modifiers. It seems that there are no concise guidelines, but since modifiers have no runtime impact, the most important consideration is code-readability. As a thumb-rule, I have added the usage of

modifiers (in the contracts mentioned in the audit-report) if and only if this usage reduces the code size in text:

- *In contract `SGATokenManager`, I added modifier `onlyIfRedButtonIsNotEnabled`.*
- *In contract `DataSource`, I left the `require` statement at the beginning of the function.*

Functions

Declare Visibility Modifier Before Any Custom Modifiers

According to Solidity v0.4.25 Style Guide, visibility modifier for a function should come before any custom modifiers.

For example, in **contracts/saga/SGAToken.sol:**

```
22
23      constructor(IContractAddressLocator contractAddressLocator) ContractAddressLocatorHolder(contractAddressLocator) public {}
24
```

The **public** visibility modifier should come before the **ContractAddressLocatorHolder** super.

For another example, also in **contracts/saga/SGAToken.sol:**

```
31      */
32      function() payable external {
33          uint256 amount = getSGATokenManager().exchangeEthForSga(msg.sender, msg.value);
34          _mint(msg.sender, amount);
35      }
36
```

The **external** visibility modifier should come before the **payable** modifier.

UPDATE: The team has resolved most of the issues:

Partially done (external visibility modifier moved before payable modifier). As for the other suggestion:

Indeed, according to the Solidity v0.4.25 Style Guide, visibility modifier for a function should come before any custom modifiers. However:

1. *Calling the 'super' doesn't quite "qualify" as a modifier.*
2. *The Arguments for Base Constructors documentation example shows that the visibility modifier indeed appears after the 'super' and not before it.*

Return Multiple Values by Name

Whenever a function returns multiple values (especially with the same type), we recommend to return them by name. This will help in returning wrong values by mistake, as well as allow to access the values by their names using web3.js 1.0 and later.

For example, in **contracts/saga/ModelDataSource.sol:**

```
72    function getIntervalCoefs(uint256 rowNum, uint256 colNum) external view returns (uint256, uint256) {
73        Interval storage interval = intervalLists[rowNum][colNum];
74        return (interval.alpha, interval.beta);
75    }
```

Can be refactored to:

```
72    function getIntervalCoefs(uint256 rowNum, uint256 colNum) external view returns (uint256 alpha, uint256 beta) {
73        Interval storage interval = intervalLists[rowNum][colNum];
74        alpha = interval.alpha;
75        beta = interval.beta;
76    }
```

UPDATE: The team has decided to keep the current style:

We decided to leave it this way because it would otherwise raise the question of why we did not apply the same in every function which returns a single value. In addition to that, the structure of such functions (assignment into variables which are not declared inside the function, no return-statement, etc) might be regarded rather unclear by many developers.

Specify Data Location for Array or Mapping Return Variables

We recommend to explicitly specify the data location of array or mapping return variables.

For example, in **contracts/utils/Adminable.sol:**

```
60    function getAdminArray() external view returns (address[]) {
61        return adminArray;
62    }
```

Can be refactored to:

```
60    function getAdminArray() external view returns (address[] memory) {
61        return adminArray;
62    }
```

UPDATE: The team has resolved the issue.

Specify Data Location for Function Parameters

We recommend specifying the data location of function parameters explicitly.

For example, in **contracts/contract_address_locator/ContractAddressLocator.sol**:

```
16    constructor(bytes32[] interfaceNames, address[] contractAddresses) public {
```

Can be refactored to:

```
16    constructor(bytes32[] memory interfaceNames, address[] memory contractAddresses) public {
```

For another example, in **contracts/utils/helpers/MultiSigWalletExposure.sol:**

```
6     constructor(address[] _owners, uint _required) MultiSigWallet(_owners, _required) public {}
```

Can be refactored to:

```
6     constructor(address[] memory _owners, uint _required) MultiSigWallet(_owners, _required) public {}
```

For another example, in **contracts/utils/helpers/MultiSigWalletTestCalls.sol:**

```
36    function receive1bytes(bytes c) setMsgFields payable public {
```

Can be refactored to:

```
36    function receive1bytes(bytes c) setMsgFields payable public {
```

UPDATE: The team has resolved the issue

Parameter Sanity Checks

Consider performing sanity checks to validate every parameter, even if when the current implementation doesn't have an execution path violating these checks. Future maintenance and modifications can easily invalidate these assumptions.

For example, in **contracts/saga/PaymentManager.sol:**

```
105   function registerPayment(address _wallet, uint256 _ethAmount) external only(_ISGATokenManager_) {
106       uint256 sdrAmount = getETHConverter().fromEthAmount(_ethAmount);
107       getPaymentQueue().addPayment(_wallet, sdrAmount);
108       emit PaymentRegistered(_wallet, _ethAmount, sdrAmount);
109   }
```

We recommend checking that the provided **wallet** isn't a 0x0 address.

UPDATE: The team has resolved the issue

Avoid Reverts in Pure or View Functions

We generally recommend avoiding reverting[5] in pure of view-only functions, as these reverts are very easy to miss/ignore when accessed via the **call()** method by clients and considering its state immutability semantics.

For example, we would recommend modifying the **approveRate** function in **contracts/saga/RateApprover.sol** to return a boolean/error enum instead and revert accordingly in **ETHConverter's setPrice** function instead.

A similar pattern is repeated when using execution preventing modifiers, but since it's very similar to assertions semantics, it should be of a much lower priority.

UPDATE: The team has resolved the issue

Cache Total Payments Sum

As a quick and safe optimization, we recommend caching the total payments sum using a state variable and dynamically updating it accordingly.

Instead of looping over the registered payments in **contracts/saga/PaymentQueue.sol:**

```
59    function getPaymentsSum() external view returns (uint256) {
60        uint256 sum = 0;
61        for (uint i = first; i < last; i++) {
62            sum = sum.add(payments[i].amount);
63        }
64        return sum;
65    }
```

We recommend adding a new **paymentsSum** state variable and using it accordingly:

```
22    Payment[] public payments;
23    uint256 public paymentsSum;
24    uint256 public first;
25    uint256 public last;
```

```
60    function getPaymentsSum() external view returns (uint256) {
61        return paymentsSum;
62    }
```

[5] Asserting, for verifying internal assumptions, is generally acceptable.

```
80      function addPayment(address _wallet, uint256 _amount) external only(_IPaymentManager_) {
81          assert(_wallet != address(0) && _amount > 0);
82          Payment memory newPayment = Payment({wallet : _wallet, amount : _amount});
83          if (payments.length > last)
84              payments[last] = newPayment;
85          else
86              payments.push(newPayment);
87          last = last.add(1);
88          paymentsSum = paymentsSum.add(_amount);
89      }
```

```
95      function updatePayment(uint256 _amount) external only(_IPaymentManager_) assertNonEmpty {
96          assert(_amount > 0);
97
98          Payment storage payment = payments[first];
99          uint256 prevAmount = payment.amount;
100
101         if (prevAmount != _amount) {
102             payment.amount = _amount;
103             paymentsSum = paymentsSum.sub(prevAmount).add(_amount);
104         }
105     }
```

```
110     function removePayment() external only(_IPaymentManager_) assertNonEmpty {
111         uint256 prevAmount = payments[first].amount;
112         delete payments[first];
113         uint256 newFirstPosition = first.add(1);
114         if (newFirstPosition == last)
115             first = last = 0;
116         else
117             first = newFirstPosition;
118         paymentsSum = paymentsSum.sub(prevAmount);
119     }
```

UPDATE: The team has resolved the issue

Conflicting Naming Conventions and Style

The project contains the following naming conventions conflicts:

Event Names

In some cases, events are named in the past tense, while in other cases - in the present tense.

For example, in **contracts/authorization/AuthorizationDataSource.sol:**

```
18      event WalletSaved(address indexed wallet);
19      event WalletDeleted(address indexed wallet);
20      event WalletNotSaved(address indexed wallet);
21      event WalletNotDeleted(address indexed wallet);
```

While in **contracts/contract_address_locator/ContractAddressLocatorProxy.sol:**

```
14      event Upgrade(IContractAddressLocator indexed prev, IContractAddressLocator indexed next);
```

In a slightly different example, in **contracts/saga/MonetaryModel.sol**, events are prefixed with the name of the contracts:

```
10  contract SagaModel is ISagaModel, ContractAddressLocatorHolder {
11      event SagaModelBuy (uint256 input, uint256 output);
12      event SagaModelSell(uint256 input, uint256 output);
```

We recommend choosing and following a single event naming convention.

UPDATE: The team has resolved the issue

Event Parameters

In some cases, even parameters are prefixed with an underscore, while in other cases - they aren't.

For example, in **contracts/saga/ReconciliationAdjuster.sol:**

```
9       event FactorSaved(uint256 _sdrFactorN, uint256 _sdrFactorD);
10      event FactorNotSaved(uint256 _sdrFactorN, uint256 _sdrFactorD);
```

While in **contracts/saga/MonetaryModel.sol:**

```
11      event SagaModelBuy (uint256 input, uint256 output);
12      event SagaModelSell(uint256 input, uint256 output);
```

We recommend choosing and following a single event parameter naming convention.

UPDATE: The team has resolved the issue

Private State Variables

In most cases, private state variables are prefixed with an underscore, while in some cases they don't use any prefix at all.

For example, in **contracts/contract_address_locator/ContractAddressLocator.sol:**

```
14      mapping(bytes32 => address) private _registry;
```

While in **contracts/saga/helpers/IntervalIteratorMockup.sol:**

```
6       uint256 private index;
7       uint256 private interval;
```

We recommend choosing and following a single private state variable naming convention.

UPDATE: The team has resolved the issue

Function Declarations

In some places, there is an extra space between the function name and its parameters.

For example, in **contracts/saga/interfaces/IInterestConverter.sol:**

```
3  interface IInterestConverter {
4      function adjustBuy (uint256 sdrAmount) external view returns (uint256);
5      function adjustSell(uint256 sdrAmount) external view returns (uint256);
6  }
```

For another example, in **contracts/saga/interfaces/ITransactionManager.sol:**

```
3  interface ITransactionManager {
4      function buy (uint256 ethAmount) external returns (uint256);
5      function sell(uint256 sgaAmount) external returns (uint256);
6  }
```

We recommend choosing and following a single function declaration notation.

UPDATE: The team has resolved the issue

Function Parameters

In some cases, function parameters are prefixed with underscores, while in most of the cases - they aren't.

For example, in **contracts/saga/ReconciliationAdjuster.sol**:

```
22      function setFactor(uint256 _sequenceNum, uint256 _sdrFactorN, uint256 _sdrFactorD) external onlyOwner {
23          require(1 <= _sdrFactorN && _sdrFactorN <= MAX_VAL);
```

While in **contracts/saga/SGAToken.sol:**

```
120     function transferEthToSgaHolder(address to, uint256 value) external only("IDebtManager") returns (uint256) {
121         uint256 amount = value < address(this).balance ? value : address(this).balance;
```

For another example, in **contracts/saga/ReserveManager.sol:**

```
24      function setWallet(address _wallet) external onlyOwner {
25          require(_wallet != address(0));
26          wallet = _wallet;
27      }
28
29      function getDepositParams(uint256 balance) external view returns (address, uint256) {
30          require(thresholds.min >= balance);
31          return (wallet, thresholds.mid - balance);
32      }
```

We recommend choosing and following a function parameter naming convention.

UPDATE: The team has resolved the issue

Indentation

In some places, the code indentation is inconsistent.

For example, in **contracts/saga/RedButton.sol:**

```
 9      function isEnabled() external view returns (bool) {
10          return enabled;
11      }
12
13      function setEnabled(bool _enabled) external onlyOwner {
14          enabled = _enabled;
15      }
```

For another example, local variables are sometimes indented to match, like in **contracts/saga/SagaModel.sol:**

```
66          sdrTotal  = minR;
67          sgaTotal  = minN;
```

While in other cases, there is no indentation at all, like in **contracts/saga/ModelCalculator.sol**:

```
28      uint256 temp = alpha - beta * sgaTotal;
29      uint256 variableFix = sdrAmount * (temp * ONE) / (temp * (ONE - DELTA) + GAMMA);
30      uint256 constantFix = sdrAmount * BUY_N / BUY_D;
```

UPDATE: The team has resolved the issue

Project and Structure

Skip Code Coverage for Test Contracts

We recommend explicitly avoiding code coverage report of helper/mock/proxy/adapter test contracts, buy specifying the **skipFiles** property in **.solcover.js.**

UPDATE: The team has resolved the issue

Unnecessary Tracked Files

We highly recommend removing/untracking **build/artifacts** from the repository. Since these files can be auto-generated from the source code, keeping them in the repository creates both additional clutter increases the chances of accidentally working with/deploying incorrect versions.

UPDATE: The team has decided to keep the auto-generated artifacts:

We decided to leave all artifacts (bin and abi files) in the repository, for the following reasons:

1. *It is actually useful in the sense that we can verify at any given moment that our source code matches our deployed code - compiling our contracts (via `npm run build`) shall reveal any mismatches because any changed artifact will appear when we check for modifications in our local repository.*
2. *Unlike executable/binary/object files, the artifacts are in essence plain text files, which makes them slightly more "legitimate" for keeping under source control (for example, we can still check for differences between different versions pretty easily).*
3. *The artifacts are indeed not relevant for our Javascript testing infrastructure, which compiles the contracts before testing them (Truffle). However, they are used in our Python testing infrastructure, which does not compile the contracts in advance. In order to allow executing these test without any preliminary actions, we need them available immediately after GIT-Pull (though this can be solved by forcing the execution of `npm run build` immediately after `npm install`).*

Missing .gitignore

It's highly recommended to have a **.gitignore** to specify which files shouldn't be committed to the repository and which files should be ignored (e.g., **npm_modules, coverage,** and so forth).

For example:

- https://github.com/ethereum/solidity-examples/blob/master/.gitignore
- https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/.gitignore

UPDATE: The team has resolved the issue.

Missing LICENSE

We recommend the team to include a detectable license in the repository so that people who visit the repository will see it at the top of the repository page.

UPDATE: The team hasn't resolved the issue yet.

Missing .node-version

A **.node-version** file will instruct Node.js version managers (e.g., avn, nodenv) which node version to use and will switch to that version of node.

UPDATE: The team has resolved the issue.

Appendix A: Audited Files

Version 2.1

Please see below the list of all the modified audited contracts and other files, followed up by this report, with their SHA256 fingerprints:

Path	SHA256 Fingerprint
contracts/authorization/AuthorizationActionRoles.sol	918d57a7c53dd62bc4951fa5b1e325ca9e50d86a806a1e65180e69b772606f44
contracts/authorization/AuthorizationDataSource.sol	0bfe2afc8d6a6a16e91809bf134f87391cf08831418f03a73865dca9738f8f46
contracts/authorization/helpers/AuthorizationDataSourceMockup.sol	21f88397096c6c3144550c33cad48ba88b2a08fb3d26fefa43bbdff32d05cbc4
contracts/authorization/interfaces/IAuthorizationDataSource.sol	9e637de3d73dc5d08e42d82ec116fb292abf042b2342e63ffea3e921e78c095b
contracts/contract_address_locator/ContractAddressLocator.sol	93368c6c5d001fd28ba0e4292271a3660b41d147ccc5c4d2c3f01cd3440748d6
contracts/contract_address_locator/ContractAddressLocatorHolder.sol	5b85edf48a546540025986dc9925d975f1b92fc1e4ca73694d1e347629028203
contracts/contract_address_locator/ContractAddressLocatorProxy.sol	7e4533ff837a4385b906aa4a38093d69b5a6e43c180b58fe5a500cbb35e16d8b
contracts/contract_address_locator/helpers/	8f3cf4d313a57dd099445daabfbc9030fd01f7b53d1dee9fe3926bd8fa9a6c96
contracts/contract_address_locator/helpers/	5e9423320cad59ecc44f21e3a3f31c2f295d28296219f792db988a2f9abc8ff7
contracts/contract_address_locator/helpers/	d5846c946ae7ceda0929781c4b2702b9b6a87611b746cd59d5696b111514293a
contracts/contract_address_locator/interfaces/	d39785beba4c70c35e7b3d07d799d0f11f18ad840be22a1cd8200f6c110ef869
contracts/Migrations.sol	d39e214137a1664cbf3fabb8e4e28555a85399ed4285b1b30305956028ce40d1
contracts/saga-genesis/helpers/MintManagerMockup.sol	f4dabfe591aa2fc31dfd59cb44e430c121b315eb75d519663c0bfaa1a7a3ead0
contracts/saga-genesis/helpers/SagaExchangerMockup.sol	f2b02c1de3fdb9928a2cd2a8bdf5e59aeff0848798687bf8be4f09926db22408
contracts/saga-genesis/helpers/SGNAuthorizationManagerMockup.sol	95f69ac17ef4144dde3e36050e1be7c2db50672b821ddb301cfd45ef67dfff75
contracts/saga-genesis/helpers/SGNConversionManagerMockup.sol	b0d6581233a1312bf35c1400ecb34b8ab34068ad9c78019c5da8fc29d5542d8d
contracts/saga-genesis/helpers/SGNTokenManagerMockup.sol	bf0dc15cd6a1f7657fb270c9ed3486a20973e00e7d62789609603543b25fc6d8
contracts/saga-genesis/interfaces/IMintHandler.sol	a6a813651a7f7e99243e4e3f4062cb0489656c676632c6d868ede390b5c7b3b5
contracts/saga-genesis/interfaces/IMintManager.sol	3f98a9a4d4b9a03ab5f7853590c27820257032f2877dbb06fcc3cd18dd6e4ed1
contracts/saga-genesis/interfaces/ISagaExchanger.sol	8c64f3168d01cb7da059385eff7a916a9e72f48dfc7882856380afae34aeb03f
contracts/saga-genesis/interfaces/ISGNAuthorizationManager.sol	b5d2b5a4dabb3e0afa55dac4db19711af3ed67b1d234772c15f2be3bd6d0abdf
contracts/saga-genesis/interfaces/ISGNConversionManager.sol	33d8afb4493504e1d5a73004dc96c530a8b130255fc6cbbcf7680a8e897055a9
contracts/saga-genesis/interfaces/ISGNTokenManager.sol	29b89c94c41f1cec19f2d45b949f35c43bf9e3a98afcf3a009c4c6b7000f75b3

contracts/saga-genesis/SGNAuthorizationManager.sol	25c3808dadd5b147a547703a291236c7ea4c060aa74af92b40d729fa80db1d94
contracts/saga-genesis/SGNConversionManager.sol	d3fd1e34fba80bc9c17230e43db15651cc4651910a6f5ce802a002e61bc560c6
contracts/saga-genesis/SGNToken.sol	8658c7f705341cd1cefc528f4071a3bfd49dc9a9e62c360f4ab4b652ec678d17
contracts/saga-genesis/SGNTokenManager.sol	48989ef47706a4e60e9e6f3a18c9a3fe59924656a4698d7cd17a49d275d82c90
contracts/saga-genesis/SGNWalletsTradingLimiter.sol	b559039ffff85d6d18cac8a3cf4cd3fc891f3521633c3c0a95f20df9e7f290d4
contracts/saga/ETHConverter.sol	95f406bf5ae4cca0ca72e5d3b7ccf31f64411c0c4fabf45da91d3c2cd575b0bc
contracts/saga/helpers/ETHConverterMockup.sol	0e707e923d80bc8d790347670a58d63389cee0d122935f50ed30118caa8fc4ad
contracts/saga/helpers/IntervalIteratorExposure.sol	e74f8a22438d0472c6d55bc82a96b7587d87e8745166420dbdf396ee57ff3d5b
contracts/saga/helpers/IntervalIteratorMockup.sol	d19459f8e110b150a93b51468f8b062def06816af61cfe69c4ea777785e27f6e
contracts/saga/helpers/MintHandlerMockup.sol	a726d1cf5d24579545a50deb026299a223c6d9e67a115762a052182a6ae93650
contracts/saga/helpers/MintingPointTimersManagerExposure.sol	1144eb52311b08e3b694ca5a07463ea7773ffafb64752866d0299bd80c2f1aa7
contracts/saga/helpers/MintingPointTimersManagerMockup.sol	8a919072e1d812ef43ad73269692d2e742edf7bab36a000f4c2d71670b946acd
contracts/saga/helpers/MintListenerMockup.sol	df502a157059a7a102f870503564949b513204b3c1b92b6524b4e82d612d9fc2
contracts/saga/helpers/MintManagerExposure.sol	4ba664889536c826c8600d4aa078cc9b229393ef900470730dc9fae4e8ce9486
contracts/saga/helpers/ModelCalculatorExposure.sol	d2e5afce939c64e2d4919919c246e1c991c9f8befdb42472cbb5a6500f5ea50f
contracts/saga/helpers/ModelCalculatorMockup.sol	cbc16627493f426404093a0264ade36776e6edd0e7f401a89aa543a1c8b58f9e
contracts/saga/helpers/ModelDataSourceMockup.sol	17da607582dc465b8e874c4694fecb09367c3558b13d11aceb4e1e20497955b4
contracts/saga/helpers/MonetaryModelMockup.sol	1fd0e56e212de4c69a4d361bcfe61fdd7ed30fc21bf645f051e4623d0f4e1268
contracts/saga/helpers/MonetaryModelStateMockup.sol	f28a55263fc60c73ee0cf0765e3c36d23a01961880b547c322b226ef36c186c6
contracts/saga/helpers/PaymentManagerMockup.sol	bb6bb684cacaac00c07f91e21e197816617e263071885911031f0c60d057309c
contracts/saga/helpers/PaymentManagerUser.sol	c00100aff0fba19d34dc17b1c3e9f91a3c28c4637a2b91824a51dc61b4d183fb
contracts/saga/helpers/PaymentQueueExposure.sol	4551c43d6927b3bffcd3b7db945b407c04d11ad270451f1a08032e1271124f06
contracts/saga/helpers/PaymentQueueMockup.sol	4c954c7cbdc284deb28b62417e4be7f82e3ae23a2a5f3abc8c9e71aef9613d51
contracts/saga/helpers/PriceBandCalculatorMockup.sol	ced874ac54ec70dee5358ddc2885181cfbc5614fa2823ba8c671e57a66278522
contracts/saga/helpers/RateApproverMockup.sol	fb6b7961d9943330152a644db56bab2eb5564cc0a43b430242b4c6e1a995050d
contracts/saga/helpers/ReconciliationAdjusterMockup.sol	5d9387e970606d3ccf4f7cf6950165abcf430fdf4d768a00f2e86f91a0795c16
contracts/saga/helpers/RedButtonMockup.sol	f2e1787a70e706d8969b5e4cc5f53cad3ea81e1360595de9a0b7ad8ee8794d8c
contracts/saga/helpers/ReserveManagerMockup.sol	6a124bba7cf94fa22606d2e514386313d54c900ac2bed72f1d21ddf014532d90
contracts/saga/helpers/SGAAuthorizationManagerMockup.sol	36720aa9a8e417ab413b4ae773be60460aeaf77772d3251bfb4fa0ecaf008852
contracts/saga/helpers/SGATokenManagerMockup.sol	42c89b421f0d0a962a70390d7cc4c55946765dc3077253da0aa5f64c0efd30cb
contracts/saga/helpers/TransactionLimiterMockup.sol	186e823a9550d0bd2d71c444f10c82a4d3b9aa1adbde8c2e6e7de0aaec552530

contracts/saga/helpers/TransactionManagerMockup.sol	8c83c51b23d4d84ad0ac5bfde5c5a9a820bb7afacc1329716a4a4710f87b3927
contracts/saga/interfaces/IETHConverter.sol	3379863cd6fee2b60f1b66c4590d0e03faea847df0f16a3b927164d14116a4a5
contracts/saga/interfaces/IIntervalIterator.sol	0921ce1af12f23b3616e450878e48110cf1b2a33ab24a14a031fafc1af9b5fb5
contracts/saga/interfaces/IMintingPointTimersManager.sol	4e1cd1e3ef36dc91de85f6dc1949cb4bfbdd479cd2e5af2904f854d3b8515bcd
contracts/saga/interfaces/IMintListener.sol	e0e2f905dcc7f36af1fcc6c191b33d36293d6e6ee376f5d3b01081294980501 5
contracts/saga/interfaces/IModelCalculator.sol	59ac15c3447126c2d53f6a539966ec3cdd04fab7512bd19c39c9a6aa560f3f8b
contracts/saga/interfaces/IModelDataSource.sol	8f6df07aecb48321abeee3805f6258b849963b86d6388da12eeb9a08b9754a59
contracts/saga/interfaces/IMonetaryModel.sol	ce46a238c27d9d4ae8e3f7990acdc2912e9e248800f5ae9da63b2be71f5bdf92
contracts/saga/interfaces/IMonetaryModelState.sol	693083387517a9470cf152dc8c9498f193492cae69e16ee52b22d433541c3785
contracts/saga/interfaces/IPaymentHandler.sol	989425288f87210846226730f7dc8f5851e7dba4110aa540ea5b778cf82c9e3c
contracts/saga/interfaces/IPaymentManager.sol	afe065e2fe1ec84fe327b78de8d0c486995cde6f5f5221fb3f81e409de9ba4a8
contracts/saga/interfaces/IPaymentQueue.sol	a855f4a902f044255afc65c86cdd91e09ca53033176c04ea85524c43b52eb449
contracts/saga/interfaces/IPriceBandCalculator.sol	2c184e26e5ea86fa39ab1686009c76d806efb7589b965f86be997da308946900
contracts/saga/interfaces/IRateApprover.sol	f230aeafd739f319dd6ceb81e41220e389181b4163dc9a7c01ba36c5b9ba8c51
contracts/saga/interfaces/IReconciliationAdjuster.sol	00a502eded28d16a1daec89d9a581bdd827cfef6c07fac68c9ce75474d517daa
contracts/saga/interfaces/IRedButton.sol	d0f2ed851346fd076e999d977dc9005f4e719da8aaacd55951ec7d5acf9d7093
contracts/saga/interfaces/IReserveManager.sol	9220b1167f3e7823836c2f80fd5e569c7c3d2ee7287d3429a0c2a0e383fb72cb
contracts/saga/interfaces/ISGAAuthorizationManager.sol	30bafa8b7e13aa37db394c6a37264bb8ac1066ac8111922d2c6711a7986fff01
contracts/saga/interfaces/ISGATokenManager.sol	a293912779a4366f1d0cb0707b74c56290ff05b7d1fdce0b982bc7d0e2b45166
contracts/saga/interfaces/ITransactionLimiter.sol	45629916811f628d47bd876aadbbbe9299455d3146cb0b8e93a01b34785805a6
contracts/saga/interfaces/ITransactionManager.sol	c873f65f3141d3bc53b9c37656e39ea44deb3e37416a98b6f4f8dada04ef3147
contracts/saga/IntervalIterator.sol	a55a40875b6e972526cacd1d43cf1ad62e865d2ecf22ea6b536e4686d95eeaab
contracts/saga/MintingPointTimersManager.sol	222031e672de1f0cb1a90eb245a1e722e31e5dba9fa9928aed3397c5cc9eebf2
contracts/saga/MintManager.sol	971561f2b39c735cb823bea52a26ccc55b33ddd3c649a5ed97a2fa03ad803217
contracts/saga/ModelCalculator.sol	876e5a83f526f239776bc6b867acd7e2258388e5606506c0487fad72045707b8
contracts/saga/ModelDataSource.sol	316e8e90e279ae4ae1223a435a903322f7d508ae2a90ea5decf776466f7e60ee
contracts/saga/MonetaryModel.sol	4806f207fc796e42aafab9c7f34907a0dda052ccab7950d35a037430dbfdf1a7
contracts/saga/MonetaryModelState.sol	a2ad5b4d9a6a1838ca87023cb9818094b0a0854b36ed851b015957baf9affb6d
contracts/saga/PaymentManager.sol	2f4b6443a4af2d6bda4bb59704d2bf23467c8461a18131077d1a142cd76da5bd
contracts/saga/PaymentQueue.sol	d1146825ca767d6dda285aa35c3cde7ea1971b9b8a7f1dab79d9220108a1169e
contracts/saga/PriceBandCalculator.sol	40cf3412b316811ef23d53a5a77b698bca293e5cdb433c3ef4539d61f5618d33

contracts/saga/RateApprover.sol	7087244d9d6133269841201f471e74a3dc9e65cf39ba837ee38cef8adf27bc91
contracts/saga/ReconciliationAdjuster.sol	28634dbf62694e018e60e7ab3907b97074c57a6613521fbc727f797745e214a6
contracts/saga/RedButton.sol	7ea94dc32e25ef63698b86d153e0a811748346e894f54a890f4ccea0909c5784
contracts/saga/ReserveManager.sol	dd6636da0844d0efc7d99d17581ad7bfda0efe1d3bb96c0f2f790be886cca15f
contracts/saga/SGAAuthorizationManager.sol	5d07ea842b4c396251af010a0249fab34eeaf5537a6b30340ef5b13a4cfc95d7
contracts/saga/SGAToken.sol	dfa87db41bf8513f210adf41c3bf74f157efcf10a978f0ca99f7bcb965342127
contracts/saga/SGATokenManager.sol	83ffaeac087c4cb74587fef29e82b3c2c558b87bf2d8ae00abb6107994be6df6
contracts/saga/SGAWalletsTradingLimiter.sol	2fdf212bf3ed329bcdadbab874a58057a27b815212cc0fb902b397a841d31316
contracts/saga/TransactionLimiter.sol	6f18a49701a77883ef566cbd38546c9c91799a656d17f854a5f63cd82d6af6fe
contracts/saga/TransactionManager.sol	a26801cbf965be2506c2071f68f187020fe670ac50f6ed0ef8698b597fe63130
contracts/utils/Adminable.sol	71dec912ec37f237dd1b91d3317859d825d93802905eda63fc04bc88d77155b5
contracts/utils/helpers/MultiSigWalletExposure.sol	ff36b1d98b6a3a80c0ad17a16b40f94ce1852facb96c79e5f25289e35f776d90
contracts/utils/helpers/MultiSigWalletTestCalls.sol	6db6908d563e6e9d98b8ae15e08a93fa7fe8142f1241e79907495e56e1ec18ca
contracts/utils/helpers/MultiSigWalletTestToken.sol	a25beef04ca7faaa384c311fa57ad397c4629a0ab252d088747005805ea39680
contracts/utils/MultiSigWallet.sol	6dc9cb05290b53509056bb8fddeb8bab30cf46e236088ebb03f2c89d69516493
contracts/wallet_trading_limiter/	63fd27f6a2e8f6c4f33d84c41ac2b03add36dd82a12bfe7b2f20eac45ed9789c
contracts/wallet_trading_limiter/helpers/	0b0b64758cc1956d3313a5e3ea88739674a1429761955b103cc78040274f727e
contracts/wallet_trading_limiter/helpers/	93e5d9a2a53e432e2289e0855d2f8b0d72d9653e2babd64c7880524c2d276671
contracts/wallet_trading_limiter/helpers/	07e16ecab0e1f4eb954f92783f2cc1727782e8121dd4cd01c84265e4657cc1a3
contracts/wallet_trading_limiter/helpers/	3a057c4f6fea7d7b215e187c0e158f40b8bb034a34958a4d5cac884f3ac24e48
contracts/wallet_trading_limiter/helpers/TradingClassesMockup.sol	5717a5e91bfe15b1dabe483dfa9ec5f526ba9b99f06d752178a06fc7c12f814d
contracts/wallet_trading_limiter/interfaces/	40afb6c2ec5736f6beb967824539349ab7bb4978334d2732220478dbea83a88b
contracts/wallet_trading_limiter/interfaces/	66963c85ee258c6337dceebb2ae041ce9bfa7c1e0711779a3c65f5e34463a6ad
contracts/wallet_trading_limiter/interfaces/ITradingClasses.sol	dccf10e9b3b4971ada2c1c50b629f926cbad763f2016c7c7dae4bc599b30e050
contracts/wallet_trading_limiter/interfaces/IWalletsTradingLimiter.sol	052cdd7e298654a3c584114df5b8e5e92a31882b5f7c5f50e2d6c3c12dbf9e40
contracts/wallet_trading_limiter/TradingClasses.sol	2294a0476c71944854ec6a3b9c14e06b64b5b0bef9592d748baa21df9a4750fd
contracts/wallet_trading_limiter/WalletsTradingDataSource.sol	cafba0a07f07bd9d46f89dba958485577824c148d3fcd2cfe4bcef6426b0b48b
contracts/wallet_trading_limiter/WalletsTradingLimiterBase.sol	e8873a27710a3d43ce31a5a6abb84978b7fc07d46272cea61f7dbe07b8d69241
migrations/1_initial_migration.js	533b7003b8ea3b5249aa48706cb4078fc6c04ee4936436f17a550f36cd979ae3
migrations/2_deploy_contracts_phase1.js	f081d2e615ed4a76168dd9175782c7153ac3b5d4c634bce9f53dd5b0fecf8369
migrations/3_deploy_contracts_phase2.js	f56b72f640c44f8a900a44daf337a65ed355105b962ba6b653b319842a76d46b

migrations/4_initialize_model_data_source.js	446319eda92d3896db1d18c38dfb1e60d5199ee54f1ff271ac078e9c1a23c548
migrations/util.js	92da65cbb5b7320709ab8cd022da5aa3429c7744d5135211efee3c6185d6e6e8
package.json	3281a3d303840bc00e66c33923c6003f14f9028871101de127aceebfabafaf3d
scripts/2_deploy_contracts_phase1.js	838163f1dc46d04efb6b75c6bc60561ef032b06bff3dcf940e7859fe52500134
scripts/3_deploy_contracts_phase2.js	a3871c9d89de4e3e6f01c61bf4a22f6b19f4c0441af81a907a53865f6e55def8
scripts/4_initialize_model_data_source.js	ff19e33f1f4fc55c24f8f8637c2c887fee68442fc5ff2a076359df4e4ab4e83d
scripts/fix-modules.js	a2f479c0c63540c0a2c97e1309176e39c1f741b858c0b0298164931c4618a956
scripts/multisig/common.js	bb6a9943e2a3506f011773e618132189715241cf396ef1e167ca846e510e6916
scripts/multisig/confirm_transaction.js	7569ea8457135389584b2b583c2f06bd5a8ebb9d638b1d548785769ca48a1320
scripts/multisig/execute_transaction.js	24e9480bdde5c0b51cfe14fe329c3a4b58ac9de2c5390f1da2f1f40745689ed6
scripts/multisig/revoke_confirmation.js	68c1c6f4f1a402d4dea285c16357d5adf23bfd09ecfe04d4b319bc38791db588
scripts/multisig/submit_transaction.js	874ea1fa91a4ac521cae552281295f78ba6258d2ada6e8f221298f2dd4702fbd
scripts/multisig/transfer_ownership.js	42be9a6b1766a0423c2ec98a47b2a8433a6c863f2b22b344f87a43c9335ee0a2
scripts/price-update/get_sdr_price.js	dcad41d6546f7ebc65d6ba92118792ba14084c16f100f5717b9609c56624ad3f
scripts/price-update/update_trading_converter.js	a848343e0753c6db3caeef898030a8eb3c9ce6c8983e87b737e306a7d9b94ccf
scripts/price-update/update_transaction_converter.js	7657d239977abfcba493b6cd95acce79d533006942100ad5d9353a9323eac124
scripts/rebuild-all.js	a48d5958595e153e852f810b6cce5475411b15dad068ef820dcbd00984f1db3b
scripts/run-tests.js	b2e1eb78e9fbc96f7e8f91516e54f5ac9348ce258e606d5f7803121413d061c3
scripts/util.js	42643558f57a3f8afb3eb8a60be31cb2457868872086997326d1e3eea071957f
scripts/verify.js	b2d10bd897c229f989884b73ab02f57c3d6e7a63eb5fdcfa63c249de9b05feca
test/authorization/AuthorizationDataSourceUnitTest.js	2bd7da788cd8c738a834c51be729ac28c9b2ff5b9d7af37b0b9b37fc2cfbc80e
test/authorization/helpers/AuthorizationActionRoles.js	1cb0824001eb6357993aef2fa48c1d5d28760a8ee1aa74a9f4635c363fab8f14
test/contract_address_locator/ContractAddressLocatorHolderUnitTest.js	fcfe09fc0c7d909255ec22a3259124da4bea0235346d5c00f82068c19ebbfbf7
test/contract_address_locator/ContractAddressLocatorProxyUnitTest.js	1a0c4465588f8646c11859f4fb85c23b6b166e2bb4de0713c7b6753159e902f8
test/contract_address_locator/ContractAddressLocatorUnitTest.js	f27f68eaf9d0d9aa6f34f77722a544f8184c17dafbae496624e5d52fe206da42
test/exceptions.js	89eff46ad85c873ec1115790fb95933fe25ba52f5a6b2a53f73c1c9fc6eb866e
test/saga-genesis/helpers/SGNConversionManager.js	c7dbac3dfa36bef32b7e186e1563c97335c11606907d6340eaa3d9c89188c836
test/saga-genesis/helpers/SGNToken.js	3b047422cb04ad4090769e1179d4852399e5e8535358e8ec79157deaeaccf9ec
test/saga-genesis/SGNAuthorizationManagerUnitTest.js	41496b2cddbd657eb9e9cf05d92531d949eff1c26054cdcf9aa12a335027311d
test/saga-genesis/SGNConversionManagerUnitTest.js	74447553369adb773fca21a27a8aa7ed7c7b8c6139d0583e3ff47fb9c3c1df4f
test/saga-genesis/SGNTokenFuncTest.js	e0545eccb69764f334feff94c3ab353adae85327f4c02fe7a8964a165323c9db

test/saga-genesis/SGNTokenManagerUnitTest.js	6025232c4f88c5faa55b053c37e5ed85bf2d9d4ee3facd5515923847b5866f03
test/saga-genesis/SGNTokenUnitTest.js	0799f8ec97fbe19ad9c013c23471cc82672973420410673e076ef5651c392ec4
test/saga-genesis/SGNWalletsTradingLimiterUnitTest.js	2efaaf8c03345bb03af8572e400d1a75379f838dcfa983798d843a9b2a955e02
test/saga/ETHConverterUnitTest.js	087487b52b58c717e9eaa8c818bf9939d9b40ac10fe530b71e4aadd95e76349d
test/saga/helpers/ModelDataSource.js	4ce47135b298ad5d033802fc3210d36a5551867172dbc905a4b5861ae5903ca7
test/saga/helpers/sequences/BuyAll.js	b6b3fff279300230be629274d3a867fac4789d0a8ba61d515579ef7a5ae1031c
test/saga/helpers/sequences/BuyAllSellAll.js	79f3194b49fd8c23db1a93018476979033da93b8ca08b642daec9b3b8ed95f9e
test/saga/helpers/sequences/BuySomeSellSome.js	e27b9d420937b9f6048294bf3a2651e9fda5f8c245cf2b2bff11864a55303d58
test/saga/helpers/ShareHolderTestSequence.js	6af91539f86fb63c5f101a82eabea86bab53b62bbf63f358ca8f00aa3ecd1315
test/saga/IntervalIteratorUnitTest.js	7949576ae61b0715f5e42ec51fadb0f2052fda81241fab667b9ed62c990550df
test/saga/MintingPointTimersManagerUnitTest.js	85e3a5957f7ee9f9713d676fb14f57aae84ac163cbcd4deebdb5c7f0de2ae430
test/saga/MintManagerUnitTest.js	c1da24a05a34048222b2c49407a88b678ad497892bc3309beb8bbb111c4779f6
test/saga/ModelCalculatorExtUnitTest.js	eb04f3d74676efbb098d7c2854021d480c004a1f570b53a9c98610e8fbce9dcd
test/saga/ModelCalculatorIntUnitTest.js	928e5035e80804643f59f56f58b0f74aeb5dd66094afa23a08d778cd2e5dfae0
test/saga/ModelDataSourceUnitTest.js	5a514c7f7dadaeb62e3ddcc3dfd61f8f4a5176555fb33dc99473644ba143d9ed
test/saga/MonetaryModelFuncTest.js	7de42240eb91daa0ada3a57b2b087e87d242426b527afadf9ffbb64ae43dda2a
test/saga/MonetaryModelStateUnitTest.js	52f187a44b0f0fe8a1bf99c5abd29e896d0fc0d1af2c16ac82bd6f74125cfe92
test/saga/MonetaryModelUnitTest.js	a4d383f5f0e3dd9a9cd04ac4ed6363a675a578fca72414036d71d435865e8ad6
test/saga/PaymentManagerUnitTest.js	fd8cef51683f6ba4939a0c0812e7a736c5979819765680fd402032b1101aff08
test/saga/PaymentQueueUnitTest.js	ae4dd2276ee85593d3215465f5bcbb60ffb74e05a3ad538df4349b6079ffda54
test/saga/PriceBandCalculatorExtUnitTest.js	2760aaf8d03d6eaea92b61fc215ec6614c56f577dd368fbe17c89842adac42be
test/saga/PriceBandCalculatorIntUnitTest.js	078daedad32474dfcaddc2c069c9e36066ba97271e674f951ddd32b192400215
test/saga/RateApproverUnitTest.js	9643dc32724dfd361b63df3621037bae29bc7a51a578eae13fbca41ab76a2ebb
test/saga/ReconciliationAdjusterUnitTest.js	86a31978bdefa56063591ade4707f178d3a808721f36b7ece116d8bf5121c6e6
test/saga/RedButtonUnitTest.js	c93b89b51a5c865622fc777afec39c48ea165f12f0f04edade385df6449dd14b
test/saga/ReserveManagerUnitTest.js	6078d38d603d5964fedf25e0983632b928ebb15ba324a9b41acd108f124b955f
test/saga/SGAAuthorizationManagerUnitTest.js	7d535e12d414cd4e33878c444dbc81153a28c2be1701914db669b5483ce7444d
test/saga/SGATokenFuncTest.js	d9f5c6ab6f88c101d629095531f87bbd2ef28a32f91a549d83166f609ff6995c
test/saga/SGATokenManagerUnitTest.js	035ed02aa2818635fbe98f8f5db5731c094bf8bf39d90d878fa791182b1dbe6b
test/saga/SGATokenUnitTest.js	f67cb350f977ec66b9e14f62753a59625ee9b4ffc504a22b253e5e1fe5ea7e5e
test/saga/SGAWalletsTradingLimiterUnitTest.js	54e5c185cdf2a87ff21099193d48e0af5c9e10d39023b86bbec7f037ec72dca8

test/saga/ShareHolderTest.js	73600dc65b27a92b10d23f3a20e3f28c46f9ccae8d2d46b14cbbe6a84add1ac5
test/saga/SystemTest.js	f2f11aa87b05855d02a5059949107fa441ab3d5b039110fd5a91b12227a2ec59
test/saga/TransactionLimiterUnitTest.js	116c88a2a35310a6332272d4d6af2e9f72f3515e13861f8bfa8797e483a97c8c
test/saga/TransactionManagerUnitTest.js	d8bfea01d09a6f6d17fb69b5b23b277d169076d23da87f1f5c6e1724b7197ee0
test/utilities.js	8e0d4ef0621645f0f2029367870dbb61564008c01b091b84a253fa05b0172134
test/utils/AdminableUnitTest.js	14aef8c8920bb402c59eeae15d22359cd006be20b004f4169184207bcd84dccd
test/utils/helpers/utilsMultiSig.js	de83e0755e229ab0a65242629069d3f206cf7c222400ae2556a885a649e89176
test/utils/MultiSigWalletTestExecutionAfterRequirementsChanged.js	d3d896c3312bf9238db16a99894bfc30a6206a63646d53511b6404cbaa719054
test/utils/MultiSigWalletTestExternalCalls.js	35d824070bd38cb446698387e0ff780b314b53ca284beb521e62cfb884d44ac5
test/utils/MultiSigWalletTestFallback.js	aae77e933fdc15621d6f2b662b7b8c24e2650c3f348186c1dd6a9c80b59f1cc2
test/utils/MultiSigWalletTestOwnership.js	7e21c0581d2ecc3ad76e3da0603180f3b8e52fd17a9cfc3ef7bbe227344ec4f6
test/utils/MultiSigWalletTestRequirements.js	7b863ac6eaa6747874a2dfb409065b6fed52aee758ec30353bd2c635be1e4df6
test/utils/MultiSigWalletTestRevocation.js	39b628ca3daf5b1defa8d008878b2d81efc55dc7f27cf4396ce923be88e18864
test/utils/MultiSigWalletTestSequence.js	c2faa3f456b009876424f23a76b6cbadd9cfdd8a030e8d752174c7e1f0b147ce
test/wallet_trading_limiter/	a8f0d9bc4a07051c549b996269d14c6a73573ed052e0f89d7e8b3599a78f8e82
test/wallet_trading_limiter/TradingClassesUnitTest.js	a47ad4784960f18a0d718ac92a1ecacc53203ea7b6bf8f00ffe934c82d4b1d47
test/wallet_trading_limiter/WalletsTradingDataSourceUnitTest.js	36f28a38597222af44d8a19e521e91daa35fc2de8a9252e06783bee4b79abc4a
truffle-config.js	dad238dcd964909131O5ff5cca9201aa68d2d5c2059aa7553f93169f59cc3889
.solcover.js	df0eb7e9c2f1e3f62e19256ce973410b44fd7f873d7105b2b772376ad6ee7ecc

Version 2.0

Please see below the list of all the modified audited contracts and other files, followed up by this report, with their SHA256 fingerprints:

Path	SHA256 Fingerprint
contracts/authorization/AuthorizationActionRoles.sol	918d57a7c53dd62bc4951fa5b1e325ca9e50d86a806a1e65180e69b772606f44
contracts/authorization/AuthorizationDataSource.sol	0bfe2afc8d6a6a16e91809bf134f87391cf08831418f03a73865dca9738f8f46
contracts/authorization/helpers/AuthorizationDataSourceMockup.sol	21f88397096c6c3144550c33cad48ba88b2a08fb3d26fefa43bbdff32d05cbc4
contracts/authorization/interfaces/IAuthorizationDataSource.sol	9e637de3d73dc5d08e42d82ec116fb292abf042b2342e63ffea3e921e78c095b
contracts/contract_address_locator/ContractAddressLocator.sol	93368c6c5d001fd28ba0e4292271a3660b41d147ccc5c4d2c3f01cd3440748d6
contracts/contract_address_locator/ContractAddressLocatorHolder.sol	a790f3daaa148864d49f6b59f3c926d534735b7b42f08a32121073107a9d7591
contracts/contract_address_locator/ContractAddressLocatorProxy.sol	7e4533ff837a4385b906aa4a38093d69b5a6e43c180b58fe5a500cbb35e16d8b
contracts/contract_address_locator/helpers/ContractAddressLocatorHolderExposure.sol	d5846c946ae7ceda0929781c4b2702b9b6a87611b746cd59d5696b111514293a
contracts/contract_address_locator/helpers/ContractAddressLocatorMockup.sol	8f3cf4d313a57dd099445daabfbc9030fd01f7b53d1dee9fe3926bd8fa9a6c96
contracts/contract_address_locator/helpers/ContractAddressLocatorProxyMockup.sol	5e9423320cad59ecc44f21e3a3f31c2f295d28296219f792db988a2f9abc8ff7
contracts/contract_address_locator/interfaces/IContractAddressLocator.sol	d39785beba4c70c35e7b3d07d799d0f11f18ad840be22a1cd8200f6c110ef869
contracts/Migrations.sol	d39e214137a1664cbf3fabb8e4e28555a85399ed4285b1b30305956028ce40d1
contracts/saga-genesis/helpers/MintManagerMockup.sol	f4dabfe591aa2fc31dfd59cb44e430c121b315eb75d519663c0bfaa1a7a3ead0
contracts/saga-genesis/helpers/SagaExchangerMockup.sol	f2b02c1de3fdb9928a2cd2a8bdf5e59aeff0848798687bf8be4f09926db22408
contracts/saga-genesis/helpers/SGNAuthorizationManagerMockup.sol	95f69ac17ef4144dde3e36050e1be7c2db50672b821ddb301cfd45ef67dfff75
contracts/saga-genesis/helpers/SGNConversionManagerMockup.sol	b0d6581233a1312bf35c1400ecb34b8ab34068ad9c78019c5da8fc29d5542d8d
contracts/saga-genesis/helpers/SGNTokenManagerMockup.sol	bf0dc15cd6a1f7657fb270c9ed3486a20973e00e7d62789609603543b25fc6d8
contracts/saga-genesis/interfaces/IMintHandler.sol	a6a813651a7f7e99243e4e3f4062cb0489656c676632c6d868ede390b5c7b3b5
contracts/saga-genesis/interfaces/IMintManager.sol	3f98a9a4d4b9a03ab5f7853590c27820257032f2877dbb06fcc3cd18dd6e4ed1
contracts/saga-genesis/interfaces/ISagaExchanger.sol	8c64f3168d01cb7da059385eff7a916a9e72f48dfc7882856380afae34aeb03f
contracts/saga-genesis/interfaces/ISGNAuthorizationManager.sol	b5d2b5a4dabb3e0afa55dac4db19711af3ed67b1d234772c15f2be3bd6d0abdf
contracts/saga-genesis/interfaces/ISGNConversionManager.sol	33d8afb4493504e1d5a73004dc96c530a8b130255fc6cbbcf7680a8e897055a9
contracts/saga-genesis/interfaces/ISGNTokenManager.sol	29b89c94c41f1cec19f2d45b949f35c43bf9e3a98afcf3a009c4c6b7000f75b3
contracts/saga-genesis/SGNAuthorizationManager.sol	25c3808dadd5b147a547703a291236c7ea4c060aa74af92b40d729fa80db1d94
contracts/saga-genesis/SGNConversionManager.sol	d3fd1e34fba80bc9c17230e43db15651cc4651910a6f5ce802a002e61bc560c6
contracts/saga-genesis/SGNToken.sol	8658c7f705341cd1cefc528f4071a3bfd49dc9a9e62c360f4ab4b652ec678d1

	7
contracts/saga-genesis/SGNTokenManager.sol	48989ef47706a4e60e9e6f3a18c9a3fe59924656a4698d7cd17a49d275d82c90
contracts/saga-genesis/SGNWalletsTradingLimiter.sol	f518959cff8ee4ab19c5cb75c68fdd19b409dc51df6719dcc44faac1ff6960c2
contracts/saga/ETHConverter.sol	d6a67557001b14dd612bdcb858d1edb2be3c0f0bb768f6b3eefefde63c097e43
contracts/saga/helpers/ETHConverterMockup.sol	0e707e923d80bc8d790347670a58d63389cee0d122935f50ed30118caa8fc4ad
contracts/saga/helpers/IntervalIteratorExposure.sol	e74f8a22438d0472c6d55bc82a96b7587d87e8745166420dbdf396ee57ff3d5b
contracts/saga/helpers/IntervalIteratorMockup.sol	d19459f8e110b150a93b51468f8b062def06816af61cfe69c4ea777785e27f6e
contracts/saga/helpers/MintHandlerMockup.sol	a726d1cf5d24579545a50deb026299a223c6d9e67a115762a052182a6ae93650
contracts/saga/helpers/MintingPointTimersManagerExposure.sol	1144eb52311b08e3b694ca5a07463ea7773ffafb64752866d0299bd80c2f1aa7
contracts/saga/helpers/MintingPointTimersManagerMockup.sol	8a919072e1d812ef43ad73269692d2e742edf7bab36a000f4c2d71670b946acd
contracts/saga/helpers/MintListenerMockup.sol	df502a157059a7a102f870503564949b513204b3c1b92b6524b4e82d612d9fc2
contracts/saga/helpers/MintManagerExposure.sol	4ba664889536c826c8600d4aa078cc9b229393ef900470730dc9fae4e8ce9486
contracts/saga/helpers/ModelCalculatorExposure.sol	d2e5afce939c64e2d4919919c246e1c991c9f8befdb42472cbb5a6500f5ea50f
contracts/saga/helpers/ModelCalculatorMockup.sol	cbc16627493f426404093a0264ade36776e6edd0e7f401a89aa543a1c8b58f9e
contracts/saga/helpers/ModelDataSourceMockup.sol	17da607582dc465b8e874c4694fecb09367c3558b13d11aceb4e1e20497955b4
contracts/saga/helpers/PaymentManagerMockup.sol	fa3070581c46b988c41e959541d2384b17fcfe57746a304379aa2721182aa65a
contracts/saga/helpers/PaymentManagerUser.sol	3c10af6291c16d37b7b4f55eb4e49a1cf016fe598a78f2178c6988ed0066ad1b
contracts/saga/helpers/PaymentQueueExposure.sol	4551c43d6927b3bffcd3b7db945b407c04d11ad270451f1a08032e1271124f06
contracts/saga/helpers/PaymentQueueMockup.sol	2dec6b18cd5838913908a4d3f54398b1431a488c0849fd91a6f17acf2bb62b94
contracts/saga/helpers/PriceBandCalculatorMockup.sol	ced874ac54ec70dee5358ddc2885181cfbc5614fa2823ba8c671e57a66278522
contracts/saga/helpers/RateApproverMockup.sol	153eb948697c822e06af72a89b25197e516b4078641f328890a56bb9cb2437ea
contracts/saga/helpers/ReconciliationAdjusterMockup.sol	5d9387e970606d3ccf4f7cf6950165abcf430fdf4d768a00f2e86f91a0795c16
contracts/saga/helpers/RedButtonMockup.sol	f2e1787a70e706d8969b5e4cc5f53cad3ea81e1360595de9a0b7ad8ee8794d8c
contracts/saga/helpers/ReserveManagerMockup.sol	6a124bba7cf94fa22606d2e514386313d54c900ac2bed72f1d21ddf014532d90
contracts/saga/helpers/SagaModelMockup.sol	b8d4cbd235b26b0c963ed729b7199d9c9d63e8917ae3a1e0810e4684df2c5322
contracts/saga/helpers/SagaModelStateMockup.sol	d8be74639a239011b9de145cf041d07715e3349f11512e1c999ebadfa9cf8fdc
contracts/saga/helpers/SGAAuthorizationManagerMockup.sol	36720aa9a8e417ab413b4ae773be60460aeaf77772d3251bfb4fa0ecaf008852
contracts/saga/helpers/SGATokenManagerMockup.sol	42c89b421f0d0a962a70390d7cc4c55946765dc3077253da0aa5f64c0efd30cb
contracts/saga/helpers/TransactionLimiterMockup.sol	186e823a9550d0bd2d71c444f10c82a4d3b9aa1adbde8c2e6e7de0aaec552530
contracts/saga/helpers/TransactionManagerMockup.sol	8c83c51b23d4d84ad0ac5bfde5c5a9a820bb7afacc1329716a4a4710f87b3927
contracts/saga/interfaces/IETHConverter.sol	3379863cd6fee2b60f1b66c4590d0e03faea847df0f16a3b927164d14116a4a5
contracts/saga/interfaces/IIntervalIterator.sol	0921ce1af12f23b3616e450878e48110cf1b2a33ab24a14a031fafc1af9b5fb

	5
contracts/saga/interfaces/IMintingPointTimersManager.sol	4e1cd1e3ef36dc91de85f6dc1949cb4bfbdd479cd2e5af2904f854d3b8515bcd
contracts/saga/interfaces/IMintListener.sol	e0e2f905dcc7f36af1fcc6c191b33d36293d6e6ee376f5d3b01081294980501 5
contracts/saga/interfaces/IModelCalculator.sol	59ac15c3447126c2d53f6a539966ec3cdd04fab7512bd19c39c9a6aa560f3f8b
contracts/saga/interfaces/IModelDataSource.sol	8f6df07aecb48321abeee3805f6258b849963b86d6388da12eeb9a08b9754a59
contracts/saga/interfaces/IPaymentHandler.sol	989425288f87210846226730f7dc8f5851e7dba4110aa540ea5b778cf82c9e3c
contracts/saga/interfaces/IPaymentManager.sol	20d8e77cfbe06a424dde5717f939866f5afae944cf9f67bf24d7899224590ff0
contracts/saga/interfaces/IPaymentQueue.sol	a855f4a902f044255afc65c86cdd91e09ca53033176c04ea85524c43b52eb449
contracts/saga/interfaces/IPriceBandCalculator.sol	2c184e26e5ea86fa39ab1686009c76d806efb7589b965f86be997da308946900
contracts/saga/interfaces/IRateApprover.sol	49065d77897560fe7c41e8097410ac0087be9153a89d09d55f1e18890ea8617e
contracts/saga/interfaces/IReconciliationAdjuster.sol	00a502eded28d16a1daec89d9a581bdd827cfef6c07fac68c9ce75474d517daa
contracts/saga/interfaces/IRedButton.sol	d0f2ed851346fd076e999d977dc9005f4e719da8aaacd55951ec7d5acf9d7093
contracts/saga/interfaces/IReserveManager.sol	9220b1167f3e7823836c2f80fd5e569c7c3d2ee7287d3429a0c2a0e383fb72cb
contracts/saga/interfaces/ISagaModel.sol	96819b668b166c5ad2f093182903769d82b4058de1e5b436f5ed0fc20483dbf7
contracts/saga/interfaces/ISagaModelState.sol	712862e1f1a7ea9935a78a56baaa1e415bf866a36243d1b9e585d5d189c93dc0
contracts/saga/interfaces/ISGAAuthorizationManager.sol	30bafa8b7e13aa37db394c6a37264bb8ac1066ac8111922d2c6711a7986fff01
contracts/saga/interfaces/ISGATokenManager.sol	a293912779a4366f1d0cb0707b74c56290ff05b7d1fdce0b982bc7d0e2b45166
contracts/saga/interfaces/ITransactionLimiter.sol	45629916811f628d47bd876aadbbbe9299455d3146cb0b8e93a01b34785805a6
contracts/saga/interfaces/ITransactionManager.sol	c873f65f3141d3bc53b9c37656e39ea44deb3e37416a98b6f4f8dada04ef3147
contracts/saga/IntervalIterator.sol	c8ff64ef161338c65e23699c825e1cb9887116e715f47e51388eb8196b70f586
contracts/saga/MintingPointTimersManager.sol	222031e672de1f0cb1a90eb245a1e722e31e5dba9fa9928aed3397c5cc9eebf2
contracts/saga/MintManager.sol	971561f2b39c735cb823bea52a26ccc55b33ddd3c649a5ed97a2fa03ad803217
contracts/saga/ModelCalculator.sol	876e5a83f526f239776bc6b867acd7e2258388e5606506c0487fad72045707b8
contracts/saga/ModelDataSource.sol	316e8e90e279ae4ae1223a435a903322f7d508ae2a90ea5decf776466f7e60ee
contracts/saga/PaymentManager.sol	57acb58b7c7c53feb39b0f5c0ee737a125c52103adeda38fc520a5a1abcbbe53
contracts/saga/PaymentQueue.sol	13728b85d1a0ee81d525c4a2d5f57aa25d370468bd5d93f38fcf68376b7127ad
contracts/saga/PriceBandCalculator.sol	40cf3412b316811ef23d53a5a77b698bca293e5cdb433c3ef4539d61f5618d33
contracts/saga/RateApprover.sol	d0ef67d65b89561650687f8c3bbe00c55d5a9e9eaff7021172784b7b4f42577e
contracts/saga/ReconciliationAdjuster.sol	28634dbf62694e018e60e7ab3907b97074c57a6613521fbc727f797745e214a6
contracts/saga/RedButton.sol	0034d0ccf53b4b86d47f05dbd531ddc251cfe6f2facddec115d20050b143b5a7
contracts/saga/ReserveManager.sol	65690fc4124398b71fd21d9028da25570616c9d8cce04c7b8ec97bfa5b3c6558
contracts/saga/SagaModel.sol	5c9fed4ca0bfb59e41934171bc8d234eab1d3476331ca612d47ba46d49b221e

	2
contracts/saga/SagaModelState.sol	83920e08df5f683f0a092f181cef5c3c0e5d5a877efb82ebb210899227426db1
contracts/saga/SGAAuthorizationManager.sol	5d07ea842b4c396251af010a0249fab34eeaf5537a6b30340ef5b13a4cfc95d7
contracts/saga/SGAToken.sol	dfa87db41bf8513f210adf41c3bf74f157efcf10a978f0ca99f7bcb965342127
contracts/saga/SGATokenManager.sol	765dcca24da636b1e62bfd406239fc0d422fb272b1d3aa56ea2226702729d7c1
contracts/saga/SGAWalletsTradingLimiter.sol	2fdf212bf3ed329bcdadbab874a58057a27b815212cc0fb902b397a841d31316
contracts/saga/TransactionLimiter.sol	8e1d59322eac26b6c91af9bfe601472176d14504a05da49520202150c53a5b8b
contracts/saga/TransactionManager.sol	1fb25912f6e9f6af0331592c0c2261cd5c9fc571bfc7299e0d90dd7131f921fc
contracts/utils/Adminable.sol	71dec912ec37f237dd1b91d3317859d825d93802905eda63fc04bc88d77155b5
contracts/utils/helpers/MultiSigWalletExposure.sol	ff36b1d98b6a3a80c0ad17a16b40f94ce1852facb96c79e5f25289e35f776d90
contracts/utils/helpers/MultiSigWalletTestCalls.sol	6db6908d563e6e9d98b8ae15e08a93fa7fe8142f1241e79907495e56e1ec18ca
contracts/utils/helpers/MultiSigWalletTestToken.sol	a25beef04ca7faaa384c311fa57ad397c4629a0ab252d088747005805ea39680
contracts/utils/MultiSigWallet.sol	6dc9cb05290b53509056bb8fddeb8bab30cf46e236088ebb03f2c89d69516493
contracts/wallet_trading_limiter/helpers/TradingClassesMockup.sol	5717a5e91bfe15b1dabe483dfa9ec5f526ba9b99f06d752178a06fc7c12f814d
contracts/wallet_trading_limiter/helpers/WalletsTradingDataSourceExposure.sol	3a057c4f6fea7d7b215e187c0e158f40b8bb034a34958a4d5cac884f3ac24e48
contracts/wallet_trading_limiter/helpers/WalletsTradingDataSourceMockup.sol	0b0b64758cc1956d3313a5e3ea88739674a1429761955b103cc78040274f727e
contracts/wallet_trading_limiter/helpers/WalletsTradingLimiterMockup.sol	93e5d9a2a53e432e2289e0855d2f8b0d72d9653e2babd64c7880524c2d276671
contracts/wallet_trading_limiter/helpers/WalletsTradingLimiterValueConverterMockup.sol	07e16ecab0e1f4eb954f92783f2cc1727782e8121dd4cd01c84265e4657cc1a3
contracts/wallet_trading_limiter/interfaces/ITradingClasses.sol	dccf10e9b3b4971ada2c1c50b629f926cbad763f2016c7c7dae4bc599b30e050
contracts/wallet_trading_limiter/interfaces/IWalletsTradingDataSource.sol	66963c85ee258c6337dceebb2ae041ce9bfa7c1e0711779a3c65f5e34463a6ad
contracts/wallet_trading_limiter/interfaces/IWalletsTradingLimiter.sol	052cdd7e298654a3c584114df5b8e5e92a31882b5f7c5f50e2d6c3c12dbf9e40
contracts/wallet_trading_limiter/interfaces/IWalletsTradingLimiterValueConverter.sol	40afb6c2ec5736f6beb967824539349ab7bb4978334d2732220478dbea83a88b
contracts/wallet_trading_limiter/TradingClasses.sol	2294a0476c71944854ec6a3b9c14e06b64b5b0bef9592d748baa21df9a4750fd
contracts/wallet_trading_limiter/WalletsTradingDataSource.sol	cafba0a07f07bd9d46f89dba958485577824c148d3fcd2cfe4bcef6426b0b48b
contracts/wallet_trading_limiter/WalletsTradingLimiterBase.sol	e8873a27710a3d43ce31a5a6abb84978b7fc07d46272cea61f7dbe07b8d69241
contracts/wallet_trading_limiter/WalletsTradingLimiterValueConverter.sol	63fd27f6a2e8f6c4f33d84c41ac2b03add36dd82a12bfe7b2f20eac45ed9789c
coverage/lcov-report/prettify.js	67126b6cd4d1b2305f8c8fa5974971ebe90ab2b0f6e209ba2f1c6e4af05f0207
coverage/lcov-report/sorter.js	a2e1ee8eb42ae6152ffb680f1f3419cf4a189412b4ffc663252492d47a968914
coverage/prettify.js	67126b6cd4d1b2305f8c8fa5974971ebe90ab2b0f6e209ba2f1c6e4af05f0207
coverage/sorter.js	a2e1ee8eb42ae6152ffb680f1f3419cf4a189412b4ffc663252492d47a968914
migrations/1_initial_migration.js	533b7003b8ea3b5249aa48706cb4078fc6c04ee4936436f17a550f36cd979ae

	3
migrations/2_deploy_contracts_phase1.js	f081d2e615ed4a76168dd9175782c7153ac3b5d4c634bce9f53dd5b0fecf8369
migrations/3_deploy_contracts_phase2.js	2512a1a99d5eef817b3b3a9dc2d0a664e765c3b40faf5a7ec07e26868c1272fe
migrations/4_initialize_model_data_source.js	446319eda92d3896db1d18c38dfb1e60d5199ee54f1ff271ac078e9c1a23c548
migrations/util.js	92da65cbb5b7320709ab8cd022da5aa3429c7744d5135211efee3c6185d6e6e8
package.json	3281a3d303840bc00e66c33923c6003f14f9028871101de127aceebfabafaf3d
scripts/2_deploy_contracts_phase1.js	838163f1dc46d04efb6b75c6bc60561ef032b06bff3dcf940e7859fe52500134
scripts/3_deploy_contracts_phase2.js	7580d3b72c11789b072366c1911abaa19808caa87103d6cfc871c77066741aa2
scripts/4_initialize_model_data_source.js	ff19e33f1f4fc55c24f8f8637c2c887fee68442fc5ff2a076359df4e4ab4e83d
scripts/fix-modules.js	a2f479c0c63540c0a2c97e1309176e39c1f741b858c0b0298164931c4618a956
scripts/multisig/common.js	bb6a9943e2a3506f011773e618132189715241cf396ef1e167ca846e510e6916
scripts/multisig/confirm_transaction.js	7569ea8457135389584b2b583c2f06bd5a8ebb9d638b1d548785769ca48a1320
scripts/multisig/execute_transaction.js	24e9480bdde5c0b51cfe14fe329c3a4b58ac9de2c5390f1da2f1f40745689ed6
scripts/multisig/revoke_confirmation.js	68c1c6f4f1a402d4dea285c16357d5adf23bfd09ecfe04d4b319bc38791db588
scripts/multisig/submit_transaction.js	874ea1fa91a4ac521cae552281295f78ba6258d2ada6e8f221298f2dd4702fbd
scripts/multisig/transfer_ownership.js	42be9a6b1766a0423c2ec98a47b2a8433a6c863f2b22b344f87a43c9335ee0a2
scripts/price-update/get_sdr_price.js	dcad41d6546f7ebc65d6ba92118792ba14084c16f100f5717b9609c56624ad3f
scripts/price-update/update_trading_converter.js	f43512041f23f09d2ea3408a0be171835b949365bd7b8e026a5a0eccafe4f00d
scripts/price-update/update_transaction_converter.js	7657d239977abfcba493b6cd95acce79d533006942100ad5d9353a9323eac124
scripts/rebuild-all.js	a48d5958595e153e852f810b6cce5475411b15dad068ef820dcbd00984f1db3b
scripts/run-tests.js	b2e1eb78e9fbc96f7e8f91516e54f5ac9348ce258e606d5f7803121413d061c3
scripts/util.js	42643558f57a3f8afb3eb8a60be31cb24578688720869973 26d1e3eea071957f
scripts/verify.js	b2d10bd897c229f989884b73ab02f57c3d6e7a63eb5fdcfa63c249de9b05feca
test/authorization/AuthorizationDataSourceUnitTest.js	2bd7da788cd8c738a834c51be729ac28c9b2ff5b9d7af37b0b9b37fc2cfbc80e
test/authorization/helpers/AuthorizationActionRoles.js	1cb0824001eb6357993aef2fa48c1d5d28760a8ee1aa74a9f4635c363fab8f14
test/contract_address_locator/ContractAddressLocatorHolderUnitTest.js	fcfe09fc0c7d909255ec22a3259124da4bea0235346d5c00f82068c19ebbfbf7
test/contract_address_locator/ContractAddressLocatorProxyUnitTest.js	1a0c4465588f8646c11859f4fb85c23b6b166e2bb4de0713c7b6753159e902f8
test/contract_address_locator/ContractAddressLocatorUnitTest.js	f27f68eaf9d0d9aa6f34f77722a544f8184c17dafbae496624e5d52fe206da42
test/exceptions.js	89eff46ad85c873ec1115790fb95933fe25ba52f5a6b2a53f73c1c9fc6eb866e
test/saga-genesis/helpers/SGNConversionManager.js	c7dbac3dfa36bef32b7e186e1563c97335c11606907d6340eaa3d9c89188c836
test/saga-genesis/helpers/SGNToken.js	3b047422cb04ad4090769e1179d4852399e5e8535358e8ec79157deaeaccf9ec
test/saga-genesis/SGNAuthorizationManagerUnitTest.js	41496b2cddbd657eb9e9cf05d92531d949eff1c26054cdcf9aa12a335027311

	d
test/saga-genesis/SGNConversionManagerUnitTest.js	74447553369adb773fca21a27a8aa7ed7c7b8c6139d0583e3ff47fb9c3c1df4f
test/saga-genesis/SGNTokenFuncTest.js	e0545eccb69764f334feff94c3ab353adae85327f4c02fe7a8964a165323c9db
test/saga-genesis/SGNTokenManagerUnitTest.js	6025232c4f88c5faa55b053c37e5ed85bf2d9d4ee3facd5515923847b5866f03
test/saga-genesis/SGNTokenUnitTest.js	0799f8ec97fbe19ad9c013c23471cc826729734204106 73e076ef5651c392ec4
test/saga-genesis/SGNWalletsTradingLimiterUnitTest.js	4f160f5132f62f8875d37afa0dfc29fe90100e6dcd03d1f5a3197c55c80908cf
test/saga/ETHConverterUnitTest.js	087487b52b58c717e9eaa8c818bf9939d9b40ac10fe530b71e4aadd95e76349d
test/saga/helpers/ModelDataSource.js	4ce47135b298ad5d033802fc3210d36a5551867172dbc905a4b5861ae5903ca7
test/saga/helpers/sequences/BuyAll.js	b6b3fff279300230be629274d3a867fac4789d0a8ba61d515579ef7a5ae1031c
test/saga/helpers/sequences/BuyAllSellAll.js	79f3194b49fd8c23db1a93018476979033da93b8ca08b642daec9b3b8ed95f9e
test/saga/helpers/sequences/BuySomeSellSome.js	e27b9d420937b9f6048294bf3a2651e9fda5f8c245cf2b2bff11864a55303d58
test/saga/helpers/ShareHolderTestSequence.js	6af91539f86fb63c5f101a82eabea86bab53b62bbf63f358ca8f00aa3ecd1315
test/saga/IntervalIteratorUnitTest.js	5628b3bbaa55c49225a5372386180ba246eee4e88dc1f8652d8b1440e5dda75a
test/saga/MintingPointTimersManagerUnitTest.js	85e3a5957f7ee9f9713d676fb14f57aae84ac163cbcd4deebdb5c7f0de2ae430
test/saga/MintManagerUnitTest.js	c1da24a05a34048222b2c49407a88b678ad497892bc3309beb8bbb111c4779f6
test/saga/ModelCalculatorExtUnitTest.js	eb04f3d74676efbb098d7c2854021d480c004a1f570b53a9c98610e8fbce9dcd
test/saga/ModelCalculatorIntUnitTest.js	928e5035e80804643f59f56f58b0f74aeb5dd66094afa23a08d778cd2e5dfae0
test/saga/ModelDataSourceUnitTest.js	5a514c7f7dadaeb62e3ddcc3dfd61f8f4a5176555fb33dc99473644ba143d9ed
test/saga/PaymentManagerUnitTest.js	05522e840091b64cd8a6182771e3d3690511baf4bfd6786a31a75e10f0e478d0
test/saga/PaymentQueueUnitTest.js	60ebb3d7c976539195648626ae2e6e3aeb90185f3b33f66c0aad14a4c31ceacf
test/saga/PriceBandCalculatorExtUnitTest.js	2760aaf8d03d6eaea92b61fc215ec6614c56f577dd368fbe17c89842adac42be
test/saga/PriceBandCalculatorIntUnitTest.js	078daedad32474dfcaddc2c069c9e36066ba97271e674f951ddd32b192400215
test/saga/RateApproverUnitTest.js	06fc3b505e81222987a91ad0866a75cd3af714c2dad38b159d4321d2ec48e2ca
test/saga/ReconciliationAdjusterUnitTest.js	86a31978bdefa56063591ade4707f178d3a808721f36b7ece116d8bf5121c6e6
test/saga/RedButtonUnitTest.js	a9a7bc6a4a29da73378751791c6061d30e5922bd5b47c1b603cadeec3f7c1d84
test/saga/ReserveManagerUnitTest.js	674b7e5bcc28a0e5d440a7d08dbd48ed0213d123b173d2609494390cca9746d4
test/saga/SagaModelFuncTest.js	93272a3a54cad5165f757adf3a8eb61cd248d2af2f4c303dc9d9bebcbe800a63
test/saga/SagaModelStateUnitTest.js	e78dfb94cd23c24b1587d0429eead17e955d576dd826d416e62051360de373e8
test/saga/SagaModelUnitTest.js	7e8af8c9eec52c161f7cfb0f391bf598942a81bed00dd33f32a28ad8d0f63317
test/saga/SGAAuthorizationManagerUnitTest.js	7d535e12d414cd4e33878c444dbc81153a28c2be1701914db669b5483ce7444d
test/saga/SGATokenFuncTest.js	f6d63a245d2e94010cebce8e4bad03d97b705cf0189a204e3cde2a768c4ad14a
test/saga/SGATokenManagerUnitTest.js	035ed02aa2818635fbe98f8f5db5731c094bf8bf39d90d878fa791182b1dbe6

	b
test/saga/SGATokenUnitTest.js	f67cb350f977ec66b9e14f62753a59625ee9b4ffc504a22b253e5e1fe5ea7e5e
test/saga/SGAWalletsTradingLimiterUnitTest.js	54e5c185cdf2a87ff21099193d48e0af5c9e10d39023b86bbec7f037ec72dca8
test/saga/ShareHolderTest.js	d9018c750d42202081aaa6636438b28c2590fc59741f4e39a456231a00652b3a
test/saga/SystemTest.js	9bfc39398354f47f85c21650f57320c7fdd7d2c2895e7fc43b66442ad0ec43a1
test/saga/TransactionLimiterUnitTest.js	4b19c724afbcaf91ed21f2484fc8b8d4b91e18df923d77d9806e9110bad627b7
test/saga/TransactionManagerUnitTest.js	abfbeb363ddeeb3941e3d985e7e9b25daf956cba9b243a9638525331a4afd722
test/utilities.js	8e0d4ef0621645f0f2029367870dbb61564008c01b091b84a253fa05b0172134
test/utils/AdminableUnitTest.js	14aef8c8920bb402c59eeae15d22359cd006be20b004f4169184207bcd84dccd
test/utils/helpers/utilsMultiSig.js	de83e0755e229ab0a65242629069d3f206cf7c222400ae2556a885a649e89176
test/utils/MultiSigWalletTestExecutionAfterRequirementsChanged.js	d3d896c3312bf9238db16a99894bfc30a6206a63646d53511b6404cbaa719054
test/utils/MultiSigWalletTestExternalCalls.js	35d824070bd38cb446698387e0ff780b314b53ca284beb521e62cfb884d44ac5
test/utils/MultiSigWalletTestFallback.js	aae77e933fdc15621d6f2b662b7b8c24e2650c3f348186c1dd6a9c80b59f1cc2
test/utils/MultiSigWalletTestOwnership.js	7e21c0581d2ecc3ad76e3da0603180f3b8e52fd17a9cfc3ef7bbe227344ec4f6
test/utils/MultiSigWalletTestRequirements.js	7b863ac6eaa6747874a2dfb409065b6fed52aee758ec30353bd2c635be1e4df6
test/utils/MultiSigWalletTestRevocation.js	39b628ca3daf5b1defa8d008878b2d81efc55dc7f27cf4396ce923be88e18864
test/utils/MultiSigWalletTestSequence.js	c2faa3f456b009876424f23a76b6cbadd9cfdd8a030e8d752174c7e1f0b147ce
test/wallet_trading_limiter/TradingClassesUnitTest.js	a47ad4784960f18a0d718ac92a1ecacc53203ea7b6bf8f00ffe934c82d4b1d47
test/wallet_trading_limiter/WalletsTradingDataSourceUnitTest.js	36f28a38597222af44d8a19e521e91daa35fc2de8a9252e06783bee4b79abc4a
test/wallet_trading_limiter/WalletsTradingLimiterValueConverterUnitTest.js	a8f0d9bc4a07051c549b996269d14c6a73573ed052e0f89d7e8b3599a78f8e82
truffle-config.js	dad238dcd96490913105ff5cca9201aa68d2d5c2059aa7553f93169f59cc3889
.solcover.js	0709a97fcef6f79f811d8a5634508920ade6642e16ce4208c0318113c31cb722

Version 1.2

Please see below the list of all the modified audited contracts and other files, followed up by this report, with their SHA256 fingerprints:

Path	SHA256 Fingerprint
contracts/authorization/AuthorizationActionRoles.sol	918d57a7c53dd62bc4951fa5b1e325ca9e50d86a806a1e65180e69b772606f44
contracts/authorization/AuthorizationDataSource.sol	53c8fe6977f8d911788c62d30f295dc0814063ac5eb768691c2344ad4698b850
contracts/authorization/helpers/AuthorizationDataSourceMockup.sol	e5aeaba5b432578c204decf00ee3b77359da27914172974478e9986441e2613d
contracts/authorization/interfaces/IAuthorizationDataSource.sol	9e637de3d73dc5d08e42d82ec116fb292abf042b2342e63ffea3e921e78c095b
contracts/Migrations.sol	d39e214137a1664cbf3fabb8e4e28555a85399ed4285b1b30305956028ce40d1
contracts/registry/ContractAddressLocator.sol	9d0b16b65201c7df301269c8e0d7d0a2e385bdd833e321c12cba6ee15b188e58
contracts/registry/ContractAddressLocatorHolder.sol	276c02391b579d70c7c192f0cdd50af628b578b6adc5b4022960768ca5fee849
contracts/registry/ContractAddressLocatorProxy.sol	210907667efae7c2b65313ee580f1a4b764fe2178d787383062f910ba7792aa9
contracts/registry/helpers/ContractAddressLocatorHolderExposure.sol	9404bf1c18a8eb70e6f7e43b8c327d4dfb101dce516410ab481695a7d4d4989d
contracts/registry/helpers/ContractAddressLocatorMockup.sol	d43cf622e398570f2622c708ed7f6f0e2a72d9b3de4e0e26580558ff7ed35178
contracts/registry/helpers/ContractAddressLocatorProxyMockup.sol	6cf52c1e2cbe140d4d0da7d64d5d0c60aac47c98d97145b7ac5271ba841f570e
contracts/registry/interfaces/IContractAddressLocator.sol	a281b8e71c3e80586b6d30d6796102ef6ef8074d64aec70a217425717adcf0a0
contracts/saga-genesis/ConversionManager.sol	acdce7bdb8d90ed587a0786e10b2e782ee0fbccc026841a9adf98f2d78e6342f
contracts/saga-genesis/helpers/ConversionManagerMockup.sol	6268b7a46b9ebb4093ad7041461a0e800df666b8c6bb19732dd5bca2a29c73ff
contracts/saga-genesis/helpers/MintManagerMockup.sol	f4dabfe591aa2fc31dfd59cb44e430c121b315eb75d519663c0bfaa1a7a3ead0
contracts/saga-genesis/helpers/SagaExchangerMockup.sol	f2b02c1de3fdb9928a2cd2a8bdf5e59aeff0848798687bf8be4f09926db22408
contracts/saga-genesis/helpers/SGNAuthorizationManagerMockup.sol	95f69ac17ef4144dde3e36050e1be7c2db50672b821ddb301cfd45ef67dfff75
contracts/saga-genesis/helpers/SGNTokenManagerMockup.sol	bf0dc15cd6a1f7657fb270c9ed3486a20973e00e7d62789609603543b25fc6d8
contracts/saga-genesis/interfaces/IConversionManager.sol	d298d7665cce19b793e4993ecf0c1e1f93af4ff550055fef06cdb5b2166089a5
contracts/saga-genesis/interfaces/IMintHandler.sol	a6a813651a7f7e99243e4e3f4062cb0489656c676632c6d868ede390b5c7b3b5
contracts/saga-genesis/interfaces/IMintManager.sol	3f98a9a4d4b9a03ab5f7853590c27820257032f2877dbb06fcc3cd18dd6e4ed1
contracts/saga-genesis/interfaces/ISagaExchanger.sol	8c64f3168d01cb7da059385eff7a916a9e72f48dfc7882856380afae34aeb03f
contracts/saga-genesis/interfaces/ISGNAuthorizationManager.sol	b5d2b5a4dabb3e0afa55dac4db19711af3ed67b1d234772c15f2be3bd6d0abdf
contracts/saga-genesis/interfaces/ISGNTokenManager.sol	29b89c94c41f1cec19f2d45b949f35c43bf9e3a98afcf3a009c4c6b7000f75b3
contracts/saga-genesis/SGNAuthorizationManager.sol	6f5368bba3a4d267f4f1e1c4bd93e6ec266d5031bcaad82570ceecd38b5b4a8f
contracts/saga-genesis/SGNToken.sol	40fdec156f0ec50e3693464d23bcbeac7feaccc372e8234a3d05907025b5636e

contracts/saga-genesis/SGNTokenManager.sol	68c08da27d88068678cc429652df9a488e10ca38d068558d708707ff4df1d4c5
contracts/saga/DataSource.sol	7dff6ab7710baf6f0a94b933cbfc36809cb13e11e299b1208152502eecdb402a
contracts/saga/DebtManager.sol	867fad3d098ed6c1308660868ba72f01c83272e7b899717be78eea7b6a1fe533
contracts/saga/DebtQueue.sol	06994c8bc7643bc6ff731949c835b743a87601aefff9ec4db32ac1e74002274d
contracts/saga/helpers/DataSourceMockup.sol	7a4fab9de10e7c30ae0ba988abe80c5efbe7bde80716ac63d33e1e917e278bc4
contracts/saga/helpers/DebtManagerMockup.sol	9d1e465720da795d0464255760ca000d3b16894502b800d5fc8f5e1b190b31fd
contracts/saga/helpers/DebtManagerUser.sol	9984c0511e1bd7b146b5de264deaced9ea81c5df4892eb5bc3b6d3e7d041e46e
contracts/saga/helpers/DebtQueueMockup.sol	8bbe96036e6727d2b7fddb895f074bbd3419b280ba10bb90f74c6ce2be1cda4b
contracts/saga/helpers/InterestConverterMockup.sol	bc86baeb0df3a29c9ae22b6957ed2cda88611d006d13918760d96f1e4f528dff
contracts/saga/helpers/IntervalIteratorExposure.sol	e74f8a22438d0472c6d55bc82a96b7587d87e8745166420dbdf396ee57ff3d5b
contracts/saga/helpers/IntervalIteratorMockup.sol	d19459f8e110b150a93b51468f8b062def06816af61cfe69c4ea777785e27f6e
contracts/saga/helpers/MintHandlerMockup.sol	a726d1cf5d24579545a50deb026299a223c6d9e67a115762a052182a6ae93650
contracts/saga/helpers/MintListenerMockup.sol	df502a157059a7a102f870503564949b513204b3c1b92b6524b4e82d612d9fc2
contracts/saga/helpers/MintManagerExposure.sol	4ba664889536c826c8600d4aa078cc9b229393ef900470730dc9fae4e8ce9486
contracts/saga/helpers/PriceBandManagerMockup.sol	b4f174d47e39452065ec358bfbc874b6f56f89967599c2be6fbfb3d5440d39b5
contracts/saga/helpers/PriceCalculatorExposure.sol	63b9763cfae69d4ccfadd9279ea31c6f604bfdda90224e60645119473d94ee7d
contracts/saga/helpers/PriceCalculatorMockup.sol	af02da15400043b8088c2f31816f3495b3e7f7f239f63cc122477dcf455e98db
contracts/saga/helpers/RedButtonMockup.sol	f2e1787a70e706d8969b5e4cc5f53cad3ea81e1360595de9a0b7ad8ee8794d8c
contracts/saga/helpers/ReserveManagerMockup.sol	6a124bba7cf94fa22606d2e514386313d54c900ac2bed72f1d21ddf014532d90
contracts/saga/helpers/SagaModelMockup.sol	b8d4cbd235b26b0c963ed729b7199d9c9d63e8917ae3a1e0810e4684df2c5322
contracts/saga/helpers/SagaModelStateMockup.sol	d8be74639a239011b9de145cf041d07715e3349f11512e1c999ebadfa9cf8fdc
contracts/saga/helpers/SGAAuthorizationManagerMockup.sol	174812e91676fb284bc3267708f5e75c13645296f4d41e8e2198eec7d2db2ef5
contracts/saga/helpers/SGATokenManagerMockup.sol	116c5c6d674e88e93495e0863352587e61fa97c9354135c6de22dd7a38e98ecb
contracts/saga/helpers/TimeManagerExposure.sol	18ff77684b3e4a88d75c2b0e6cb549ed1c784b36a852ff0a7e7f106f24d8690c
contracts/saga/helpers/TimeManagerMockup.sol	794ef49d9fe9fc70c2cd3d94d659f4722b20bca76cc5c064c7b78dadd10ccae8
contracts/saga/helpers/TradingConverterMockup.sol	9e19e130c8990d596c42a91f23803faf0a1f62859115403c0526b39a473a6999
contracts/saga/helpers/TradingDataSourceMockup.sol	926c144647895c76ada882d1b052f82db51431826f3396bb665dcc716ae68b9d
contracts/saga/helpers/TradingManagerMockup.sol	b93fbb64fb341e09d2b6875fba15d5677a40ce3c8a41071b296f2d2f0db4a038
contracts/saga/helpers/TransactionConverterMockup.sol	60b212a068484610bcbf5a580cd0740bee09347c491b0c0b9f8eb6f68f9419c9
contracts/saga/helpers/TransactionLimiterMockup.sol	186e823a9550d0bd2d71c444f10c82a4d3b9aa1adbde8c2e6e7de0aaec552530
contracts/saga/helpers/TransactionManagerMockup.sol	8c83c51b23d4d84ad0ac5bfde5c5a9a820bb7afacc1329716a4a4710f87b3927

contracts/saga/InterestConverter.sol	ac9128b62292198500fa2f8610181aca7beff1041be1cba8f5766c28422ad0e4
contracts/saga/interfaces/IDataSource.sol	518b47d9b36811d303364db508b5457f073a7246bc96346e519c4548189ed675
contracts/saga/interfaces/IDebtHandler.sol	fc9433a8f3d4bc87850ae9bd58b3cf477d6ad7c67aa7bb5b21d2cfc2bb489c69
contracts/saga/interfaces/IDebtManager.sol	dc5132b1f720483393c74c34973bcaf4114b9b77cd305586e1ddfd167aa4b356
contracts/saga/interfaces/IDebtQueue.sol	62f65da671fdf1222810643e94d1b2e3b150b7a95b867141d5b967fd33ba4511
contracts/saga/interfaces/IInterestConverter.sol	0838bb6c51135bb061dc1f1da606c43b6878368d6d9a28fab52c7759d1f4e43e
contracts/saga/interfaces/IIntervalIterator.sol	0921ce1af12f23b3616e450878e48110cf1b2a33ab24a14a031fafc1af9b5fb5
contracts/saga/interfaces/IMintListener.sol	e0e2f905dcc7f36af1fcc6c191b33d36293d6e6ee376f5d3b010812949805015
contracts/saga/interfaces/IPriceBandManager.sol	681ca8bf8c6228ad35d7d26abb9ff9a562cdd57eb9ffead6ebc97ea610fa7f13
contracts/saga/interfaces/IPriceCalculator.sol	e315644ba90184eb796aaf7646bac6a453be1be2fa326585781901cd56ae5ee5
contracts/saga/interfaces/IRedButton.sol	d0f2ed851346fd076e999d977dc9005f4e719da8aaacd55951ec7d5acf9d7093
contracts/saga/interfaces/IReserveManager.sol	9220b1167f3e7823836c2f80fd5e569c7c3d2ee7287d3429a0c2a0e383fb72cb
contracts/saga/interfaces/ISagaModel.sol	96819b668b166c5ad2f093182903769d82b4058de1e5b436f5ed0fc20483dbf7
contracts/saga/interfaces/ISagaModelState.sol	712862e1f1a7ea9935a78a56baaa1e415bf866a36243d1b9e585d5d189c93dc0
contracts/saga/interfaces/ISGAAuthorizationManager.sol	060a7e7ea3ac0e9de44c4b86e4e3df07d5b5abed1b92e227dea054ecaaff3e9e
contracts/saga/interfaces/ISGATokenManager.sol	94b05f01a1db697648b3bad652d1df9d32548f8d6932ebd07debe2c7ba51f3a3
contracts/saga/interfaces/ITimeManager.sol	28077c0f29b2e7b3582f8769026fc32af89739a5659b0bf4eff3a38414597cc2
contracts/saga/interfaces/ITradingConverter.sol	991e167eeabb72ef2c1e0a7df795556ec0f424582b02354ba823e4476a9d894b
contracts/saga/interfaces/ITradingDataSource.sol	a7e8f4ba401fdd8df8e08fe68a1c5e4a15bb900eced31d9333b7701321a2e5dd
contracts/saga/interfaces/ITradingManager.sol	ec60169f0d88627d194faa760b3162d713160db71daa3a8639e30b9b97ed59b8
contracts/saga/interfaces/ITransactionConverter.sol	668b2d688560a73e3160ce8ef6f321c657250aec53fcf9e6de3f3aade58498d5
contracts/saga/interfaces/ITransactionLimiter.sol	45629916811f628d47bd876aadbbbe9299455d3146cb0b8e93a01b34785805a6
contracts/saga/interfaces/ITransactionManager.sol	c873f65f3141d3bc53b9c37656e39ea44deb3e37416a98b6f4f8dada04ef3147
contracts/saga/IntervalIterator.sol	af85f1614d929e35b5afcdaf856384acfc9f28aadd6faef5baedcdfaca31e276
contracts/saga/MintManager.sol	0dceb637c4a5109b1b768f957c0532136bf00d4bced9594c456f4b74934794aa
contracts/saga/PriceBandManager.sol	b1facde596b1003f4cab58f3191ad0e4789b23c9181fb24be20197680cb22ccb
contracts/saga/PriceCalculator.sol	a8a25dc095db8a60884df5c3380e2c3769add10c95c48cd9074f9d86209cbe8d
contracts/saga/RedButton.sol	0034d0ccf53b4b86d47f05dbd531ddc251cfe6f2facddec115d20050b143b5a7
contracts/saga/ReserveManager.sol	2c1ccf1b3f40fbbc82f9d768204f74d30b7a776fc67e90334ab60b92ef5e3c5c
contracts/saga/SagaModel.sol	6b2bc57857d0693b344e54f1b92b37ee79f6cf0f2f15ca2fe5762a3aece85467
contracts/saga/SagaModelState.sol	d578380a3a3b65de7705c5204ab5104d4bd6938ec8d2f01bbc00678c543a4ef0

contracts/saga/SGAAuthorizationManager.sol	b82bacba0805c43b1b22304c522b37a074b4672c16398f2d1e381e320e62da63
contracts/saga/SGAToken.sol	c44670efaf90123974c0aa043bc9ca8eb7ad624af122313c3fc911c14098634d
contracts/saga/SGATokenManager.sol	513b829cd990d33577af9ae7577c0c1fe0b5f6b7d95268fdd537330be93586b5
contracts/saga/TimeManager.sol	b1b39fce5fbb47401d7cdbd0b467a09edb0855553e0710370fc503ec13819c7a
contracts/saga/TradingConverter.sol	aa6e70ffe58c13fe62cf4cbec89d9df740dc32d89143f27b222051f5cc9d6377
contracts/saga/TradingDataSource.sol	78a4067a1d262523d11d2568a4aa4dfe8824d107a46c43345bbeac9f7ff5f153
contracts/saga/TradingManager.sol	f4c3d7a8ceade15eb2e14421fd6dec083bfc4668d866b0205a2f032d16bcc1bc
contracts/saga/TransactionConverter.sol	4a290244d26afe93249d5fac95e1626cc99b18415904b87c20ce386b3f318e9f
contracts/saga/TransactionLimiter.sol	2bdf9e3f68f4276f05543f12ff2c67bcc42ff74135f789c33dfe7e049783e5b4
contracts/saga/TransactionManager.sol	7e72bc9db263e1b44ba0dfabf43b7032bb6e7a16668800870232eb80a4179bc4
contracts/utils/Adminable.sol	71dec912ec37f237dd1b91d3317859d825d93802905eda63fc04bc88d77155b5
contracts/utils/helpers/MultiSigWalletExposure.sol	ff36b1d98b6a3a80c0ad17a16b40f94ce1852facb96c79e5f25289e35f776d90
contracts/utils/helpers/MultiSigWalletTestCalls.sol	6db6908d563e6e9d98b8ae15e08a93fa7fe8142f1241e79907495e56e1ec18ca
contracts/utils/helpers/MultiSigWalletTestToken.sol	a25beef04ca7faaa384c311fa57ad397c4629a0ab252d088747005805ea39680
contracts/utils/MultiSigWallet.sol	6585f715ff51580e213898c753131978d3c83b7bd3dfdef9195d91bb90662c6c
migrations/1_initial_migration.js	533b7003b8ea3b5249aa48706cb4078fc6c04ee4936436f17a550f36cd979ae3
migrations/2_deploy_contracts_phase1.js	86c75776e5e2173a546f1b8b31191f2104121e58cc9597b0b75df6764235e348
migrations/3_deploy_contracts_phase2.js	1f29c71458f8f01c06ed0af295a11fceb6460efea7824ed3f71e43d748318494
migrations/4_initialize_data_source.js	5a3c48386c9d7a384086c61439862d0fae26763443d9ccccd997df0f4af3f615
migrations/util.js	a3755a0ec179d3d80a4c00b5fc95c9b7dca186926d33770149a0fe42d5cee496
scripts/2_deploy_contracts_phase1.js	fc2ca6bdba531868f144c8c2dfe9e2a22948e14bc0815ea6363c8dc8a4fbd4e3
scripts/3_deploy_contracts_phase2.js	1a4191e6d6f25f58a9f036e3d84400946d3aca0d3bdf721db7d092515a52207a
scripts/4_initialize_data_source.js	cc96c868c20104455ae7ac523b94f4edb37d712ff993ccde31cbf0a8a7b09db1
scripts/multisig/all/claim_ownership.confirm.js	af38c43583ebdad39db95d4c9d3c799c164c229c5d53c4639b09aa36dd1c5395
scripts/multisig/all/claim_ownership.submit.js	f59a9c1274acd66e5829771f6624a3446fb251617bc0a6d5f3663fc04e888474
scripts/multisig/all/transfer_ownership.js	eaa9439d5c248b36052d88f6896a9df10922889fbb2996400b17f1a07c2ffcb5
scripts/multisig/authorization-data-source/accept_admin.confirm.js	33bb92a8d572980bf45bc5ceff1bc578e29860e2fca390c53e1a86906dcd1d1e
scripts/multisig/authorization-data-source/accept_admin.submit.js	95ec3812bf79b9625f606fdce4801b566797b06aa90db344a2efa51115f61911
scripts/multisig/authorization-data-source/reject_admin.confirm.js	33bb92a8d572980bf45bc5ceff1bc578e29860e2fca390c53e1a86906dcd1d1e
scripts/multisig/authorization-data-source/reject_admin.submit.js	b4e875f16ce06a32512bd1ff7de7be47153c29ee82f3632e4dc8f6ef94453dac
scripts/multisig/contract-address-locator-proxy/upgrade_system.confirm.js	5771eba3cb317e43b8c6357d4b163500b52d85b16fff8800dbbfee97e1769556

scripts/multisig/contract-address-locator-proxy/upgrade_system.submit.js	f6bb05fc4ed0539e430cc3405902c9539647022209ef2314a56b09d9d183f8f1
scripts/multisig/multisig.js	ba4da373633a21a34fd40d1f09cdc8342d424dd4675f964349e413b800e78892
scripts/multisig/red-button/set_enabled.confirm.js	948a31da38a522c0671ea35b5f9c80a926a91aa00099646e1e5af3f94fd92b3e
scripts/multisig/red-button/set_enabled.submit.js	dfb55425c69975a0972abda0a3c5ae8e127f2bd7df24fd8602358ed74675e57d
scripts/multisig/reserve-manager/set_thresholds.confirm.js	93416e626241c92b6ac1f8a752b0b88c65521b3b0f99f816bc419077a8d20724
scripts/multisig/reserve-manager/set_thresholds.submit.js	35de3c69114fa90cc14dd969c0bd39d3b2396dbb95e43cee119c460a9e4339ec
scripts/multisig/reserve-manager/set_wallets.confirm.js	d9955ffe19aaf9644eb7b86d1d16c99b888b937b830ec51924510e15d9027b8f
scripts/multisig/reserve-manager/set_wallets.submit.js	c74cebf11d0d9d1667d19d722b83e1c629c721ab0d238a5c8b95852d4d7b254b
scripts/multisig/transaction-limiter/set_max_diff.confirm.js	67eded98591003873bbd4626dd62fee9a1f240421a23c05885f821bd8546cf18
scripts/multisig/transaction-limiter/set_max_diff.submit.js	a80aa003d1bb574ab7d0aaed2c498d8555fdb5ab0e4b1567aed5e0dfc98b7e82
scripts/price-update/get_sdr_price.js	dcad41d6546f7ebc65d6ba92118792ba14084c16f100f5717b9609c56624ad3f
scripts/price-update/update_trading_converter.js	a117d7746845935aee05de2175ad959df6d8c8d46b4fadbadf3e69c8236ab877
scripts/price-update/update_transaction_converter.js	3651330fe38590ccb2b4b2a624a7239645e1552f4df4f66dcfe2c259eb992279
scripts/util.js	eb02a56bf8b6d4dbcd66317523e7f337a3cce21accc81a51e1e9d30dc1bae723
test/authorization/AuthorizationDataSourceUnitTest.js	85b03dd1ca19c35d774632c859d555db1c4f30f178a57f6a4a2ca651acf9867a
test/authorization/helpers/AuthorizationActionRoles.js	1cb0824001eb6357993aef2fa48c1d5d28760a8ee1aa74a9f4635c363fab8f14
test/exceptions.js	89eff46ad85c873ec1115790fb95933fe25ba52f5a6b2a53f73c1c9fc6eb866e
test/registry/ContractAddressLocatorHolderUnitTest.js	11444d6806f1921ad7ebfcf075340f866c8fa2d84590b0cab9dfac0b2d51cb1e
test/registry/ContractAddressLocatorProxyUnitTest.js	fae66c864e26ad6d50a732e5f7612e32f28b5bcd2774fffabc0541a52eb575be
test/registry/ContractAddressLocatorUnitTest.js	faaeb7a723e45e3d503005ae39018088c2d7a3d4f81b7a8a971d468d59db27fd
test/saga-genesis/ConversionManagerUnitTest.js	71867eb6c86c519f8ebb0ced2bd078bd5ec52c7d8d0d9ce61e81647c4724d4a1
test/saga-genesis/helpers/ConversionManager.js	c7dbac3dfa36bef32b7e186e1563c97335c11606907d6340eaa3d9c89188c836
test/saga-genesis/helpers/SGNToken.js	3b047422cb04ad4090769e1179d4852399e5e8535358e8ec79157deaeaccf9ec
test/saga-genesis/SGNAuthorizationManagerUnitTest.js	41496b2cddbd657eb9e9cf05d92531d949eff1c26054cdcf9aa12a335027311d
test/saga-genesis/SGNTokenFuncTest.js	7b7c91899ac50584a49cac3ccdb7b5895227cb96ef84878eede6fa1a519a4d00
test/saga-genesis/SGNTokenManagerUnitTest.js	b13a38ba9c935b7ea941b5e9bac421bc2e7629f0ea69498db235d93f5a046de7
test/saga-genesis/SGNTokenUnitTest.js	0799f8ec97fbe19ad9c013c23471cc82672973420410673e076ef5651c392ec4
test/saga/DataSourceUnitTest.js	e0d08bc49b95a25402126323ef0469b21ee799fe1fb2911ca050a30f444c292d
test/saga/DebtManagerUnitTest.js	1e98befec8b103a9e7066c1f007b0c194d95524ecc75f5364c2016259d713bdc
test/saga/DebtQueueUnitTest.js	7e9dcafaec79576cb05fbbb69caa4a1e1ec6a5c78530be322fcb543a5211c30e
test/saga/helpers/DataSource.js	9f655cbfb5156432f827f4f6b1195295bff24da52a4ae9086fb83c5c8c76f88a

test/saga/helpers/sequences/BuyAll.js	b6b3fff279300230be629274d3a867fac4789d0a8ba61d515579ef7a5ae1031c
test/saga/helpers/sequences/BuyAllSellAll.js	79f3194b49fd8c23db1a93018476979033da93b8ca08b642daec9b3b8ed95f9e
test/saga/helpers/sequences/BuySomeSellSome.js	e27b9d420937b9f6048294bf3a2651e9fda5f8c245cf2b2bff11864a55303d58
test/saga/helpers/ShareHolderTestSequence.js	6af91539f86fb63c5f101a82eabea86bab53b62bbf63f358ca8f00aa3ecd1315
test/saga/InterestConverterUnitTest.js	8862e208ed79032347db1cd9babaa75bcbbbc0af33d84dfa741a13fe37067d90
test/saga/IntervalIteratorUnitTest.js	5ee36e26108c7d019b797aae63100e035eaa5bc0603e72186b111389a3d64f51
test/saga/MintManagerUnitTest.js	3a35afc27ad39b8525b94336d14109477344760c70cefffa678f0e146dfdd78
test/saga/PriceBandManagerExtUnitTest.js	17d181a58de9a59e9c146889271d55440326688694534 22f26c91f879c275ab1
test/saga/PriceBandManagerIntUnitTest.js	75cb3c372565534c95ee117f75bf3b9b7eb712c919c1f1fda4cdee703600e3e8
test/saga/PriceCalculatorExtUnitTest.js	22c2f67b2ee63dd265aecf595a88c08128457fa08dc868ff7c4e305241020c19
test/saga/PriceCalculatorIntUnitTest.js	1155b7a77f8c633a3dcad8ac0c2a5d236a835e66a8065f7497ebf0cadfd04f5f
test/saga/RedButtonUnitTest.js	a9a7bc6a4a29da73378751791c6061d30e5922bd5b47c1b603cadeec3f7c1d84
test/saga/ReserveManagerUnitTest.js	970a759f202c82e3523d7bca2f587cbf415b5f42c0a5cac4a63fd7e67e9300d9
test/saga/SagaModelFuncTest.js	8452132389fa11e66980a7ac949a530003210c68a685a4d443075974e7901ef7
test/saga/SagaModelStateUnitTest.js	e78dfb94cd23c24b1587d0429eead17e955d576dd826d416e62051360de373e8
test/saga/SagaModelUnitTest.js	e73f5630e2756bad2ea64d3a94c2979d7209299f67fbfa28164e998f9de78df4
test/saga/SGAAuthorizationManagerUnitTest.js	2583d2a449698f30c287d02f1350e3cf5a48fb3a049e2a2003951426ed074edf
test/saga/SGATokenFuncTest.js	b3d425478808cd1e80a3f57f35954c6a8dedbd895f2fbf9f5606142e0b8f53bb
test/saga/SGATokenManagerUnitTest.js	b8ae3bc06a4b316262975cc6927327d6962972f64d44d75445ff7cd4eada1ce8
test/saga/SGATokenUnitTest.js	83fd525702e4b29b4a2d36dfe86acc8a50b45f353afaf87c428b365f6d7bd05d
test/saga/ShareHolderTest.js	e86e52e6deeb1d26abcd6d1b5cde62ad25be5c7e0b172b5ee43df8ced3baea89
test/saga/SystemTest.js	984ef1c18dc82f2e24f6062fba6801b17bbe30a77134beb72ed7487b31de5cc3
test/saga/TimeManagerUnitTest.js	07f19519c9e24aced4f9859d7aff0ecd341879e418706536ba75641321dec2a9
test/saga/TradingConverterUnitTest.js	7b47aadee6ef697692aaaa074667c402345d32bcbc79ca3756cda7f557163c86
test/saga/TradingDataSourceUnitTest.js	864c2b30a8cbb0b359c318b27f5ce326331266d1e56d3b89a8334f165b5308db
test/saga/TradingManagerUnitTest.js	35eb5ebd7e2b6ace8a6ebce942cf875b9bf5b08e8af442e52fa05691ffafecb4
test/saga/TransactionConverterUnitTest.js	b9fcb76f5597471b4e0c09e1bc22bce8cd5b2e1ab38a07eae47202017864da59
test/saga/TransactionLimiterUnitTest.js	6a6372203c5fda14238b94397737006e7511c662da71a499f2918123ba6bb38c
test/saga/TransactionManagerUnitTest.js	537ce43f9fc42cd6389e313d1740334c09ae33af2bee1e3851a78854c182a63b
test/utilities.js	5fc712f89780f0f0fbedae3f39cc409f2c1dfb0724e644769ece510125318879
test/utils/AdminableUnitTest.js	e8f956b42a0a44491034be7dc03fdaf8acaee9c6a713a07c699bc152e819f069

test/utils/helpers/utilsMultiSig.js	de83e0755e229ab0a65242629069d3f206cf7c222400ae2556a885a649e89176
test/utils/MultiSigWalletTestExecutionAfterRequirementsChanged.js	d3d896c3312bf9238db16a99894bfc30a6206a63646d53511b6404cbaa719054
test/utils/MultiSigWalletTestExternalCalls.js	35d824070bd38cb446698387e0ff780b314b53ca284beb521e62cfb884d44ac5
test/utils/MultiSigWalletTestFallback.js	aae77e933fdc15621d6f2b662b7b8c24e2650c3f348186c1dd6a9c80b59f1cc2
test/utils/MultiSigWalletTestOwnership.js	7e21c0581d2ecc3ad76e3da0603180f3b8e52fd17a9cfc3ef7bbe227344ec4f6
test/utils/MultiSigWalletTestRequirements.js	7b863ac6eaa6747874a2dfb409065b6fed52aee758ec30353bd2c635be1e4df6
test/utils/MultiSigWalletTestRevocation.js	39b628ca3daf5b1defa8d008878b2d81efc55dc7f27cf4396ce923be88e18864
test/utils/MultiSigWalletTestSequence.js	c2faa3f456b009876424f23a76b6cbadd9cfdd8a030e8d752174c7e1f0b147ce
truffle-config.js	dad238dcd96490913105ff5cca9201aa68d2d5c2059aa7553f93169f59cc3889
.solcover.js	0709a97fcef6f79f811d8a5634508920ade6642e16ce4208c0318113c31cb722

Version 1.1

Please see below the list of all the modified audited contracts and other files, followed up by this report, with their SHA256 fingerprints:

Path	SHA256 Fingerprint
contracts/authorization/AuthorizationDataSource.sol	85eaf83a46b34b226516924ff84cd5fd975b5f213f469e62c0f124d82d1e4ed3
contracts/authorization/helpers/AuthorizationDataSourceMockup.sol	312b5ac0e0d47ca260cafddbbad4fb27caeafe131f3f5cb174166f564188d905
contracts/authorization/interfaces/IAuthorizationDataSource.sol	bc26f4bc0dcf2100f2e838e6bd6e0d42eae397b904d40a0ffbba7e859233bea3
contracts/Migrations.sol	d39e214137a1664cbf3fabb8e4e28555a85399ed4285b1b30305956028ce40d1
contracts/registry/ContractAddressLocator.sol	9d0b16b65201c7df301269c8e0d7d0a2e385bdd833e321c12cba6ee15b188e58
contracts/registry/ContractAddressLocatorHolder.sol	276c02391b579d70c7c192f0cdd50af628b578b6adc5b4022960768ca5fee849
contracts/registry/ContractAddressLocatorProxy.sol	210907667efae7c2b65313ee580f1a4b764fe2178d787383062f910ba7792aa9
contracts/registry/helpers/ContractAddressLocatorHolderExposure.sol	9404bf1c18a8eb70e6f7e43b8c327d4dfb101dce516410ab481695a7d4d4989d
contracts/registry/helpers/ContractAddressLocatorMockup.sol	d43cf622e398570f2622c708ed7f6f0e2a72d9b3de4e0e26580558ff7ed35178
contracts/registry/helpers/ContractAddressLocatorProxyMockup.sol	6cf52c1e2cbe140d4d0da7d64d5d0c60aac47c98d97145b7ac5271ba841f570e
contracts/registry/interfaces/IContractAddressLocator.sol	a281b8e71c3e80586b6d30d6796102ef6ef8074d64aec70a217425717adcf0a0
contracts/saga-genesis/ConversionManager.sol	5b7a71f7d08af1ffb2280e9eaf0fe20501bcf693f501d3a9ba43236f84a84341
contracts/saga-genesis/helpers/ConversionManagerMockup.sol	6268b7a46b9ebb4093ad7041461a0e800df666b8c6bb19732dd5bca2a29c73ff
contracts/saga-genesis/helpers/MintManagerMockup.sol	f4dabfe591aa2fc31dfd59cb44e430c121b315eb75d519663c0bfaa1a7a3ead0
contracts/saga-genesis/helpers/SagaExchangerMockup.sol	f2b02c1de3fdb9928a2cd2a8bdf5e59aeff0848798687bf8be4f09926db22408
contracts/saga-genesis/helpers/SGNAuthorizationManagerMockup.sol	f74ab2aba794fe334ab0b5519eeb3981ebfaeb91dd3fc035bbd6eb7c370a68f6
contracts/saga-genesis/helpers/SGNTokenManagerMockup.sol	bf0dc15cd6a1f7657fb270c9ed3486a20973e00e7d62789609603543b25fc6d8
contracts/saga-genesis/interfaces/IConversionManager.sol	d298d7665cce19b793e4993ecf0c1e1f93af4ff550055fef06cdb5b2166089a5
contracts/saga-genesis/interfaces/IMintHandler.sol	a6a813651a7f7e99243e4e3f4062cb0489656c676632c6d868ede390b5c7b3b5
contracts/saga-genesis/interfaces/IMintManager.sol	3f98a9a4d4b9a03ab5f7853590c27820257032f2877dbb06fcc3cd18dd6e4ed1
contracts/saga-genesis/interfaces/ISagaExchanger.sol	8c64f3168d01cb7da059385eff7a916a9e72f48dfc7882856380afae34aeb03f
contracts/saga-genesis/interfaces/ISGNAuthorizationManager.sol	0365a4fd22531473f8b1bdb6c9e72ba9cae5ee4dc0b17d4352026fdab0dac96a
contracts/saga-genesis/interfaces/ISGNTokenManager.sol	29b89c94c41f1cec19f2d45b949f35c43bf9e3a98afcf3a009c4c6b7000f75b3
contracts/saga-genesis/SGNAuthorizationManager.sol	d0371e01c4030ee9534d0b660a4a0b00faa5ed240adbd0cdf14a593ca0eccd65
contracts/saga-genesis/SGNToken.sol	40fdec156f0ec50e3693464d23bcbeac7feaccc372e8234a3d05907025b5636e
contracts/saga-genesis/SGNTokenManager.sol	68c08da27d88068678cc429652df9a488e10ca38d068558d708707ff4df1d4c5
contracts/saga/DataSource.sol	7dff6ab7710baf6f0a94b933cbfc36809cb13e11e299b1208152502eecdb402

	a
contracts/saga/DebtManager.sol	867fad3d098ed6c1308660868ba72f01c83272e7b899717be78eea7b6a1fe533
contracts/saga/DebtQueue.sol	06994c8bc7643bc6ff731949c835b743a87601aefff9ec4db32ac1e74002274d
contracts/saga/helpers/DataSourceMockup.sol	7a4fab9de10e7c30ae0ba988abe80c5efbe7bde80716ac63d33e1e917e278bc4
contracts/saga/helpers/DebtManagerMockup.sol	9d1e465720da795d0464255760ca000d3b16894502b800d5fc8f5e1b190b31fd
contracts/saga/helpers/DebtManagerUser.sol	9984c0511e1bd7b146b5de264deaced9ea81c5df4892eb5bc3b6d3e7d041e46e
contracts/saga/helpers/DebtQueueMockup.sol	8bbe96036e6727d2b7fddb895f074bbd3419b280ba10bb90f74c6ce2be1cda4b
contracts/saga/helpers/InterestConverterMockup.sol	bc86baeb0df3a29c9ae22b6957ed2cda88611d006d13918760d96f1e4f528dff
contracts/saga/helpers/IntervalIteratorExposure.sol	e74f8a22438d0472c6d55bc82a96b7587d87e8745166420dbdf396ee57ff3d5b
contracts/saga/helpers/IntervalIteratorMockup.sol	d19459f8e110b150a93b51468f8b062def06816af61cfe69c4ea777785e27f6e
contracts/saga/helpers/MintHandlerMockup.sol	a726d1cf5d24579545a50deb026299a223c6d9e67a115762a052182a6ae93650
contracts/saga/helpers/MintListenerMockup.sol	df502a157059a7a102f870503564949b513204b3c1b92b6524b4e82d612d9fc2
contracts/saga/helpers/MintManagerExposure.sol	4ba664889536c826c8600d4aa078cc9b229393ef900470730dc9fae4e8ce9486
contracts/saga/helpers/PriceBandManagerMockup.sol	b4f174d47e39452065ec358bfbc874b6f56f89967599c2be6fbfb3d5440d39b5
contracts/saga/helpers/PriceCalculatorExposure.sol	63b9763cfae69d4ccfadd9279ea31c6f604bfdda90224e60645119473d94ee7d
contracts/saga/helpers/PriceCalculatorMockup.sol	af02da15400043b8088c2f31816f3495b3e7f7f239f63cc122477dcf455e98db
contracts/saga/helpers/RedButtonMockup.sol	f2e1787a70e706d8969b5e4cc5f53cad3ea81e1360595de9a0b7ad8ee8794d8c
contracts/saga/helpers/ReserveManagerMockup.sol	6a124bba7cf94fa22606d2e514386313d54c900ac2bed72f1d21ddf014532d90
contracts/saga/helpers/SagaModelMockup.sol	b8d4cbd235b26b0c963ed729b7199d9c9d63e8917ae3a1e0810e4684df2c5322
contracts/saga/helpers/SagaModelStateMockup.sol	d8be74639a239011b9de145cf041d07715e3349f11512e1c999ebadfa9cf8fdc
contracts/saga/helpers/SGAAuthorizationManagerMockup.sol	d1e76106f0c6b7dabc23e311d382038b35417c20a1e2f928291f59778e608e34
contracts/saga/helpers/SGATokenManagerMockup.sol	116c5c6d674e88e93495e0863352587e61fa97c9354135c6de22dd7a38e98ecb
contracts/saga/helpers/TimeManagerExposure.sol	18ff77684b3e4a88d75c2b0e6cb549ed1c784b36a852ff0a7e7f106f24d8690c
contracts/saga/helpers/TimeManagerMockup.sol	794ef49d9fe9fc70c2cd3d94d659f4722b20bca76cc5c064c7b78dadd10ccae8
contracts/saga/helpers/TradingConverterMockup.sol	9e19e130c8990d596c42a91f23803faf0a1f62859115403c0526b39a473a6999
contracts/saga/helpers/TradingDataSourceMockup.sol	c87a57f7c17eed11769af62b39b93e3e582818430f9fc993d9771c5ddac84fc4
contracts/saga/helpers/TradingManagerMockup.sol	b93fbb64fb341e09d2b6875fba15d5677a40ce3c8a41071b296f2d2f0db4a038
contracts/saga/helpers/TransactionConverterMockup.sol	60b212a068484610bcbf5a580cd0740bee09347c491b0c0b9f8eb6f68f9419c9
contracts/saga/helpers/TransactionLimiterMockup.sol	186e823a9550d0bd2d71c444f10c82a4d3b9aa1adbde8c2e6e7de0aaec552530
contracts/saga/helpers/TransactionManagerMockup.sol	8c83c51b23d4d84ad0ac5bfde5c5a9a820bb7afacc1329716a4a4710f87b3927
contracts/saga/InterestConverter.sol	ac9128b62292198500fa2f8610181aca7beff1041be1cba8f5766c28422ad0e4
contracts/saga/interfaces/IDataSource.sol	518b47d9b36811d303364db508b5457f073a7246bc96346e519c4548189ed67

	5
contracts/saga/interfaces/IDebtHandler.sol	fc9433a8f3d4bc87850ae9bd58b3cf477d6ad7c67aa7bb5b21d2cfc2bb489c69
contracts/saga/interfaces/IDebtManager.sol	dc5132b1f720483393c74c34973bcaf4114b9b77cd305586e1ddfd167aa4b356
contracts/saga/interfaces/IDebtQueue.sol	62f65da671fdf1222810643e94d1b2e3b150b7a95b867141d5b967fd33ba4511
contracts/saga/interfaces/IInterestConverter.sol	0838bb6c51135bb061dc1f1da606c43b6878368d6d9a28fab52c7759d1f4e43e
contracts/saga/interfaces/IIntervalIterator.sol	0921ce1af12f23b3616e450878e48110cf1b2a33ab24a14a031fafc1af9b5fb5
contracts/saga/interfaces/IMintListener.sol	e0e2f905dcc7f36af1fcc6c191b33d36293d6e6ee376f5d3b0108129498050015
contracts/saga/interfaces/IPriceBandManager.sol	681ca8bf8c6228ad35d7d26abb9ff9a562cdd57eb9ffead6ebc97ea610fa7f13
contracts/saga/interfaces/IPriceCalculator.sol	e315644ba90184eb796aaf7646bac6a453be1be2fa326585781901cd56ae5ee5
contracts/saga/interfaces/IRedButton.sol	d0f2ed851346fd076e999d977dc9005f4e719da8aaacd55951ec7d5acf9d7093
contracts/saga/interfaces/IReserveManager.sol	9220b1167f3e7823836c2f80fd5e569c7c3d2ee7287d3429a0c2a0e383fb72cb
contracts/saga/interfaces/ISagaModel.sol	96819b668b166c5ad2f093182903769d82b4058de1e5b436f5ed0fc20483dbf7
contracts/saga/interfaces/ISagaModelState.sol	712862e1f1a7ea9935a78a56baaa1e415bf866a36243d1b9e585d5d189c93dc0
contracts/saga/interfaces/ISGAAuthorizationManager.sol	d3851531b30ed5756828bcd7dc53023ac0fe1a938ad683c8e7f5d7b54bda4f62
contracts/saga/interfaces/ISGATokenManager.sol	94b05f01a1db697648b3bad652d1df9d32548f8d6932ebd07debe2c7ba51f3a3
contracts/saga/interfaces/ITimeManager.sol	28077c0f29b2e7b3582f8769026fc32af89739a5659b0bf4eff3a38414597cc2
contracts/saga/interfaces/ITradingConverter.sol	991e167eeabb72ef2c1e0a7df795556ec0f424582b02354ba823e4476a9d894b
contracts/saga/interfaces/ITradingDataSource.sol	1705dc7bdc3d0950bf9c829c883c9ce5a047e20508e2a8dcc45d46496244ecfc
contracts/saga/interfaces/ITradingManager.sol	ec60169f0d88627d194faa760b3162d713160db71daa3a8639e30b9b97ed59b8
contracts/saga/interfaces/ITransactionConverter.sol	668b2d688560a73e3160ce8ef6f321c657250aec53fcf9e6de3f3aade58498d5
contracts/saga/interfaces/ITransactionLimiter.sol	45629916811f628d47bd876aadbbbe9299455d3146cb0b8e93a01b34785805a6
contracts/saga/interfaces/ITransactionManager.sol	c873f65f3141d3bc53b9c37656e39ea44deb3e37416a98b6f4f8dada04ef3147
contracts/saga/IntervalIterator.sol	af85f1614d929e35b5afcdaf856384acfc9f28aadd6faef5baedcdfaca31e276
contracts/saga/MintManager.sol	0dceb637c4a5109b1b768f957c0532136bf00d4bced9594c456f4b74934794aa
contracts/saga/PriceBandManager.sol	b1facde596b1003f4cab58f3191ad0e4789b23c9181fb24be20197680cb22ccb
contracts/saga/PriceCalculator.sol	19d4b17054740fb67a5d4a7252bb597fcc58752f89ab7e3cac50a8b778e6ee2a
contracts/saga/RedButton.sol	0034d0ccf53b4b86d47f05dbd531ddc251cfe6f2facddec115d20050b143b5a7
contracts/saga/ReserveManager.sol	2c1ccf1b3f40fbbc82f9d768204f74d30b7a776fc67e90334ab60b92ef5e3c5c
contracts/saga/SagaModel.sol	6b2bc57857d0693b344e54f1b92b37ee79f6cf0f2f15ca2fe5762a3aece85467
contracts/saga/SagaModelState.sol	d578380a3a3b65de7705c5204ab5104d4bd6938ec8d2f01bbc00678c543a4ef0
contracts/saga/SGAAuthorizationManager.sol	fc9ece21693fec137b44d341679e05fbfd761be5151436ae71b1db5f796b3a48
contracts/saga/SGAToken.sol	c44670efaf90123974c0aa043bc9ca8eb7ad624af122313c3fc911c14098634

	d
contracts/saga/SGATokenManager.sol	513b829cd990d33577af9ae7577c0c1fe0b5f6b7d95268fdd537330be93586b5
contracts/saga/TimeManager.sol	b1b39fce5fbb47401d7cdbd0b467a09edb0855553e0710370fc503ec13819c7a
contracts/saga/TradingConverter.sol	aa6e70ffe58c13fe62cf4cbec89d9df740dc32d89143f27b222051f5cc9d6377
contracts/saga/TradingDataSource.sol	b1576f1342e9b04d6ae2c1902d699f0eac198e0bc212e0a6979fb2c924e9ed85
contracts/saga/TradingManager.sol	4fe734dffcdb909b2b9782a458920e633bc7631074f5e080a214ebd682c4b688
contracts/saga/TransactionConverter.sol	4a290244d26afe93249d5fac95e1626cc99b18415904b87c20ce386b3f318e9f
contracts/saga/TransactionLimiter.sol	2bdf9e3f68f4276f05543f12ff2c67bcc42ff74135f789c33dfe7e049783e5b4
contracts/saga/TransactionManager.sol	7e72bc9db263e1b44ba0dfabf43b7032bb6e7a16668800870232eb80a4179bc4
contracts/utils/Adminable.sol	71dec912ec37f237dd1b91d3317859d825d93802905eda63fc04bc88d77155b5
contracts/utils/helpers/MultiSigWalletExposure.sol	ff36b1d98b6a3a80c0ad17a16b40f94ce1852facb96c79e5f25289e35f776d90
contracts/utils/helpers/MultiSigWalletTestCalls.sol	6db6908d563e6e9d98b8ae15e08a93fa7fe8142f1241e79907495e56e1ec18ca
contracts/utils/helpers/MultiSigWalletTestToken.sol	a25beef04ca7faaa384c311fa57ad397c4629a0ab252d088747005805ea39680
contracts/utils/MultiSigWallet.sol	6585f715ff51580e213898c753131978d3c83b7bd3dfdef9195d91bb90662c6c
migrations/1_initial_migration.js	533b7003b8ea3b5249aa48706cb4078fc6c04ee4936436f17a550f36cd979ae3
migrations/2_deploy_contracts_phase1.js	86c75776e5e2173a546f1b8b31191f2104121e58cc9597b0b75df6764235e348
migrations/3_deploy_contracts_phase2.js	1f29c71458f8f01c06ed0af295a11fceb6460efea7824ed3f71e43d748318494
migrations/4_initialize_data_source.js	5a3c48386c9d7a384086c61439862d0fae26763443d9ccccd997df0f4af3f615
migrations/util.js	a3755a0ec179d3d80a4c00b5fc95c9b7dca186926d33770149a0fe42d5cee496
package.json	6c1d0acd04b20027dfe339cc9021d3481b56587b4c1bc785e3f0107ffb088652
scripts/2_deploy_contracts_phase1.js	1be4772de8ffa8556e0a37f7f6096fbfb8a901b709fcb420ba2bdba52979750a
scripts/3_deploy_contracts_phase2.js	51bdc2812034a3693df3762a6b13a03ce198fc30431acdb8c73ff0c536263511
scripts/4_initialize_data_source.js	05467489ed76dad7a13847faf6f0e794fa58f4d4efea744e83b6ae61f9874069
scripts/fix-modules.js	77e7425c0f0120c99226b62588fed1bb667bdf6532cf0c4e70e756eaa5f2fc0d
scripts/multisig/all/claim_ownership.confirm.js	2ec7060c19ef52b8eac14b881c43ed759bc30328dda0c9c1b7e463432b745e1e
scripts/multisig/all/claim_ownership.submit.js	1b1608944f7e8e0523728a9cd878b6961d5fe2e4a24507ad8b5a835da41d94ec
scripts/multisig/all/transfer_ownership.js	e09231266bd4b0e19191935cb7aebb151d51b01a87ac37f49b854c74647e80be
scripts/multisig/authorization-data-source/accept_admin.confirm.js	c5d0d5e9a538381804d2462b0c851f7c2830f469ba87d7bebac60c31ca915d6d
scripts/multisig/authorization-data-source/accept_admin.submit.js	8f6133d0f32983092676aa7e687b4ab864a091672c70f3a385aae002b23792c8
scripts/multisig/authorization-data-source/reject_admin.confirm.js	c5d0d5e9a538381804d2462b0c851f7c2830f469ba87d7bebac60c31ca915d6d
scripts/multisig/authorization-data-source/reject_admin.submit.js	3502723e5cc819efadb7b525b3600989575d1907c216b793b13b69477845a432
scripts/multisig/contract-address-locator-proxy/upgrade_system.confirm.js	0157075337b54895f21d980c1591e27ee5b17f6f9e20dd2d66743514ffbabce

	2
scripts/multisig/contract-address-locator-proxy/upgrade_system.submit.js	86e03895fc679f1142288d3870dd40a45eb139bebad38b39d8ddcf292f08c955
scripts/multisig/multisig.js	ba4da373633a21a34fd40d1f09cdc8342d424dd4675f964349e413b800e78892
scripts/multisig/red-button/set_enabled.confirm.js	279fdda18ef61a63f79a4ef24c196595264f9e19fb9c3fd1612b02b8c21cc4ec
scripts/multisig/red-button/set_enabled.submit.js	af7c2b3b4237e066f8780fd9924cbc3e850716fb7627f062bdd73407bed16e18
scripts/multisig/reserve-manager/set_thresholds.confirm.js	2232c7ae34eb89313282a5daae33fc0b9aea9a0481af9439ac5245765558447f
scripts/multisig/reserve-manager/set_thresholds.submit.js	62a1dcc39ade23df3c4f6b45eaab06cd057d37613e5ee6352aa54bf48841af82
scripts/multisig/reserve-manager/set_wallets.confirm.js	324ee666e4cf15a8fc0605dfc64859874396a220dd722b36c778f9623d81f893
scripts/multisig/reserve-manager/set_wallets.submit.js	2c44d93b7e741cb251a6fd723b156550212c1760f2b13b61d9656a0c58dc85aa
scripts/multisig/transaction-limiter/set_max_diff.confirm.js	949698ccc4055f97a4a5c00f4b88bf6b28a17ca02d925179209e9085e17d2c0d
scripts/multisig/transaction-limiter/set_max_diff.submit.js	308aec42dce7db97817b93b0e2d4585f90693a6c227e4a3ae780d1244eab4bac
scripts/price-update/get_sdr_price.js	646eb91ae7cfe4975d8d7592c49d742377bc6e8a60870baee739ec8d2ceaf1c3
scripts/price-update/update_trading_converter.js	e0dc340194d2151b9b230e957241c4c9d04e83b1433590cd3cce30e817dd77e3
scripts/price-update/update_transaction_converter.js	d57c72ba91dc450822f5e5e9db8184309afcee2d4ec0e4c8db6b3f01bcd0a699
scripts/rebuild-all.js	448e2826bf9951c98744e866d75e7c02acb95be77d83dc106d5fd60cb133c309
scripts/run-tests.js	f22ab74bd940a04d6efd8db458415c26cf1ec3986d61738bd162a6063cf34b7f
scripts/util.js	95950767e353f771b660bcab93d26c8f275d2387ab4644384b3e93e0463acd04
scripts/verify.js	b2d10bd897c229f989884b73ab02f57c3d6e7a63eb5fdcfa63c249de9b05feca
test/authorization/AuthorizationDataSourceUnitTest.js	54959e735a32e87d74b02b3839ab401a3ee7d7219223f16f8353d9802c6b70b7
test/exceptions.js	89eff46ad85c873ec1115790fb95933fe25ba52f5a6b2a53f73c1c9fc6eb866e
test/registry/ContractAddressLocatorHolderUnitTest.js	53010aec94312de5f42cc64a052fea9c242b4a5a8fb425b319f1349f851f58b1
test/registry/ContractAddressLocatorProxyUnitTest.js	6e5033afd17bf69dfa2e77c16f3a351acc14f0d737b8c09e3165134c600bf3da
test/registry/ContractAddressLocatorUnitTest.js	5acb6482d641a43578d7184815120375c64c308e7947f1d95fbd20294e8980fa
test/saga-genesis/ConversionManagerUnitTest.js	71867eb6c86c519f8ebb0ced2bd078bd5ec52c7d8d0d9ce61e81647c4724d4a1
test/saga-genesis/helpers/ConversionManager.js	c7dbac3dfa36bef32b7e186e1563c97335c11606907d6340eaa3d9c89188c836
test/saga-genesis/helpers/SGNToken.js	3b047422cb04ad4090769e1179d4852399e5e8535358e8ec79157deaeaccf9ec
test/saga-genesis/SGNAuthorizationManagerUnitTest.js	b5954228d83a66638c217e3025d9e680da29490ddff8476da151fec2e0013dbe
test/saga-genesis/SGNTokenFuncTest.js	5841e08f31f06fe908f044ce67bce9bfcf7ea45dd7d0f3bd1fd0a1b034b9a794
test/saga-genesis/SGNTokenManagerUnitTest.js	a8126361b6297ad12e56353a2a2eb72ca2e556c037a5c50b370866b4b5aeeb12
test/saga-genesis/SGNTokenUnitTest.js	2b7232d808cd609c7fda3ff6150411c679591749cfeee8448bfc23ec6e9bddc9
test/saga/DataSourceUnitTest.js	e0d08bc49b95a25402126323ef0469b21ee799fe1fb2911ca050a30f444c292d
test/saga/DebtManagerUnitTest.js	85bf89480894e8a4b775fb92551431cea7f6f944151aff1917694975987cc18

	a
test/saga/DebtQueueUnitTest.js	c4712086036e28838333404aa7dbadddd9d1ca9da4e8d545a4f63d9b35e3b39f
test/saga/helpers/DataSource.js	9f655cbfb5156432f827f4f6b1195295bff24da52a4ae9086fb83c5c8c76f88a
test/saga/helpers/sequences/BuyAll.js	8bdafcd51465e8ca862a4d13282aa22bfc8e074b732d8b3badd85e447571eb25
test/saga/helpers/sequences/BuyAllSellAll.js	5957622518b3fd7c831bc130028b4a1cd0b2169584e094b0b906b6e2c5996fb4
test/saga/helpers/sequences/BuySomeSellSome.js	050ffca4f6824bf913b7b82c93e1cec48ad02b87cefa52ca38a81195ff1ab335
test/saga/helpers/ShareHolderTestSequence.js	6af91539f86fb63c5f101a82eabea86bab53b62bbf63f358ca8f00aa3ecd1315
test/saga/InterestConverterUnitTest.js	656e098db3eaf688b209865c0899cb705a1f7d12a425616afb6e69386da41afb
test/saga/IntervalIteratorUnitTest.js	5ee36e26108c7d019b797aae63100e035eaa5bc0603e72186b111389a3d64f51
test/saga/MintManagerUnitTest.js	3a35afc27ad39b8525b94336d14109477344760c70cefffa678f0e146dfdd78
test/saga/PriceBandManagerExtUnitTest.js	17d181a58de9a59e9c146889271d5544032668869453422f26c91f879c275ab1
test/saga/PriceBandManagerIntUnitTest.js	75cb3c372565534c95ee117f75bf3b9b7eb712c919c1f1fda4cdee703600e3e8
test/saga/PriceCalculatorExtUnitTest.js	2071aeb0e5315951b09f18f2843334da4f948f91efa87f10f603e09851dfe084
test/saga/PriceCalculatorIntUnitTest.js	1155b7a77f8c633a3dcad8ac0c2a5d236a835e66a8065f7497ebf0cadfd04f5f
test/saga/RedButtonUnitTest.js	a9a7bc6a4a29da73378751791c6061d30e5922bd5b47c1b603cadeec3f7c1d84
test/saga/ReserveManagerUnitTest.js	d56143fb1500fae3485243832200487d4ba9d09bf33af2f2e9480fc08416dbc3
test/saga/SagaModelFuncTest.js	8452132389fa11e66980a7ac949a530003210c68a685a4d443075974e7901ef7
test/saga/SagaModelStateUnitTest.js	e78dfb94cd23c24b1587d0429eead17e955d576dd826d416e62051360de373e8
test/saga/SagaModelUnitTest.js	e73f5630e2756bad2ea64d3a94c2979d7209299f67fbfa28164e998f9de78df4
test/saga/SGAAuthorizationManagerUnitTest.js	13ca7a6749150a1eacb0e85a49d68e1802f80c4c93ccd56fd3e503f07717ca2e
test/saga/SGATokenFuncTest.js	9aa2e2d4b579e9ed400bb34b79ff27e124abe5b50e4acaec812ba5c378b2e068
test/saga/SGATokenManagerUnitTest.js	090f04d2e508fb43a39c52fde118a485a04758794435c67b0e3546eadad0c7d2
test/saga/SGATokenUnitTest.js	f1b159f184a6ecb3d094c531bc7350612b908d1e395917b443c7eccc26b2380b
test/saga/ShareHolderTest.js	3201e143d08344847979be612b410acf6a913a9b74b0b3d3e03401102f24ad9b
test/saga/SystemTest.js	011ef85f076f2b087711dd9ed8ed56dff7a48fd9ce25da66e57ab122560455cd
test/saga/TimeManagerUnitTest.js	07f19519c9e24aced4f9859d7aff0ecd341879e418706536ba75641321dec2a9
test/saga/TradingConverterUnitTest.js	802eb344583ff497d0df17c68c8181be3967751a73c9fb493bac7f38cb52634a
test/saga/TradingDataSourceUnitTest.js	683bb4a276734aa0e043ef076a76ebd4380baed0692dbfba760ffb005c4046ff
test/saga/TradingManagerUnitTest.js	713b46968e1d55666250d70fc1ec3376fa4ba87f3c3c4a3aa8609b2bd3477029
test/saga/TransactionConverterUnitTest.js	15720142df9710a4f115b7ee0d5688a503d31b57f900b10ec042ac638780ee66
test/saga/TransactionLimiterUnitTest.js	056e51b66208d2fb1341ddef151e43c6636865c82201a404d559a58d738dc77d
test/saga/TransactionManagerUnitTest.js	537ce43f9fc42cd6389e313d1740334c09ae33af2bee1e3851a78854c182a63

	b
test/utilities.js	711e6951720fce510bc61ec08c356bfa8872a555ef91e5b01fff4e8af0b9055a
test/utils/AdminableUnitTest.js	461977f21ee42abacdcd88f1ad55f82f6d73d1638fe921d910d245917c4059c0
test/utils/helpers/utilsMultiSig.js	de83e0755e229ab0a65242629069d3f206cf7c222400ae2556a885a649e89176
test/utils/MultiSigWalletTestExecutionAfterRequirementsChanged.js	d3d896c3312bf9238db16a99894bfc30a6206a63646d53511b6404cbaa719054
test/utils/MultiSigWalletTestExternalCalls.js	35d824070bd38cb446698387e0ff780b314b53ca284beb521e62cfb884d44ac5
test/utils/MultiSigWalletTestFallback.js	2fffab9822e54608573526a81a1e95399a9802779396f534fb11660a6a6a1a5c
test/utils/MultiSigWalletTestOwnership.js	7e21c0581d2ecc3ad76e3da0603180f3b8e52fd17a9cfc3ef7bbe227344ec4f6
test/utils/MultiSigWalletTestRequirements.js	7b863ac6eaa6747874a2dfb409065b6fed52aee758ec30353bd2c635be1e4df6
test/utils/MultiSigWalletTestRevocation.js	39b628ca3daf5b1defa8d008878b2d81efc55dc7f27cf4396ce923be88e18864
test/utils/MultiSigWalletTestSequence.js	c2faa3f456b009876424f23a76b6cbadd9cfdd8a030e8d752174c7e1f0b147ce
truffle-config.js	9f926f1c4f433c33676a79820a12ba80669acda0cb16e8a954098dbf1c5aa5be
.solcover.js	0709a97fcef6f79f811d8a5634508920ade6642e16ce4208c0318113c31cb722

Version 1.0

Please see below the list of all audited contracts and other files, with their SHA256 fingerprints:

Path	SHA256 Fingerprint
contracts/authorization/AuthorizationDataSource.sol	5b33d1cd773f56265ff7823fbe2d734171a765e14212f3d79a088f63a74bd445
contracts/authorization/helpers/AuthorizationDataSourceMockup.sol	850ba163dbce339a494976bdb3753e878f6218d4f9821ab03f58cbb621e7ff79
contracts/authorization/interfaces/IAuthorizationDataSource.sol	82d1bf54f9c3c76cb746cbf8bba0630521cd07d2e7a3ee94716979107903b93c
contracts/Migrations.sol	45c062a2e7039e75c47b128c592a587e554b85aed97d52ffd9e5382219c3fe1b
contracts/registry/ContractAddressLocator.sol	7ed4cb710b8fb7a5d4572adcd9be004b341ec3c52a33d5e3d803dfc45f21878c
contracts/registry/ContractAddressLocatorHolder.sol	3e7c2d99a8dcd898cb2d66f2f6c002a5223ccf5c6eddd26ba51f5d2e2ef359ba
contracts/registry/ContractAddressLocatorProxy.sol	586c047467cfcdd41b52cb70078ab0768f64ba270fb23b5f3b779e240101d1e8
contracts/registry/helpers/ContractAddressLocatorHolderExposure.sol	ad8df21d8365ff913a0363e7edf734180d407f49e0d41de2927243afac36f5cc
contracts/registry/helpers/ContractAddressLocatorMockup.sol	46808d3dc9c00dfb65500b1a8b6e7560b44c441af7ca1c20c902a6c77d1d305c
contracts/registry/helpers/ContractAddressLocatorProxyMockup.sol	b3c0ca06d8833c0f35f64e38e6cb3e1a97d92fcd7c30d6cd7bd481c551ec34b1
contracts/registry/interfaces/IContractAddressLocator.sol	0a9de7617213102e01687bdb3a3d06df8578ff83ff1bd401f0e2d9297c9035a8
contracts/saga-genesis/ConversionManager.sol	b6c86e58729228c39e459186750681a65d495d59922785eeb3531fc8e303d3b9
contracts/saga-genesis/helpers/ConversionManagerMockup.sol	63ecd829427f67169314631b0524aaef85b0716c74a9ed682cddfc68a8d9b4e9
contracts/saga-genesis/helpers/MintManagerMockup.sol	d946703d1204e716f52cc6de12a8447600fd42ef541f8037898a45b067922840
contracts/saga-genesis/helpers/SagaExchangerMockup.sol	e65bb1d9d7ff75c75dc5b23ac6d2ef34e5053db4cc9bad5067e349dc22623674
contracts/saga-genesis/helpers/SGNAuthorizationManagerMockup.sol	0cfee3dff903495f03147df20f50cae6c44f0c3ee69734fe355e1be583a3b6b3

contracts/saga-genesis/helpers/SGNTokenManagerMockup.sol	182987ecded3404e010c3359ddb878be9c464feab1f41f0ced16bc3a99a6e7d3
contracts/saga-genesis/interfaces/IConversionManager.sol	2366408a1643fd7b7d4e4f31b6be4513ba2b427c546ebc4acb454a215bda883d
contracts/saga-genesis/interfaces/IMintManager.sol	231b4f3dd909f75ef48395533e8d2077f89988310338d020bcefc630e9fff210
contracts/saga-genesis/interfaces/ISagaExchanger.sol	7a67b7506d8f2b6f8f5be9e8f2dae6e5ff0fdcd0d4ffcbf89e391968c72f38d7
contracts/saga-genesis/interfaces/ISGNAuthorizationManager.sol	08b93d9d3fba27ff02238d5b5b00b796b2deeb08c62fd8eb4eb52cfe7f34f5ff
contracts/saga-genesis/interfaces/ISGNTokenManager.sol	18ee7d7791cb18c8a484e17a0b3cbd69e4bc2a53efe509f0b0a01bb284c97fbd
contracts/saga-genesis/SGNAuthorizationManager.sol	78b4f878958d82bc190da89d14ed592767b489acd2b0d09414b5b77956df7ec4
contracts/saga-genesis/SGNToken.sol	45aac5c59dbf0403e9ee702ebf457341da9176656006ec1f071091d8e8b54fe5
contracts/saga-genesis/SGNTokenManager.sol	6a87aa8545ff902b665c8fba03feb7128e772e66c797819dc4127f2b7766b1db
contracts/saga/DataSource.sol	39ba2a7eb6aa16b4221e84cdd9015b32dd973277e7bd4ae564a1447356d98070
contracts/saga/DebtManager.sol	9d312ddbcfbd2dc278d9c9813ab96629990ef91fe19a38fb8dc169c6dcd85ec3
contracts/saga/DebtQueue.sol	1b0f2b667724592295068046419ff4492d3b294a8f01ef4000ff1833d25de1d2
contracts/saga/helpers/DataSourceMockup.sol	1a0c8705124fa4144d3b346502aab1388a99d5c4a569a38eaf7e22203871a463
contracts/saga/helpers/DebtManagerMockup.sol	3254175ccb59f83d0cffd3b1524fbc044486e6ac58745c65006d5adcbbddf2d4
contracts/saga/helpers/DebtManagerUser.sol	b062da03ef96278adf01eba0afda6a36860ac79adef262132cb548724a63e780
contracts/saga/helpers/DebtQueueMockup.sol	246dbf3e1b70b52ad4566a2dbaf4468b9901cb85b83222e132bdd3551141d322
contracts/saga/helpers/InterestConverterMockup.sol	ef8376b316502a88b06728c1b01ead1719be5ca8fdcf968d75632493482796b7
contracts/saga/helpers/IntervalIteratorExposure.sol	be8fb9eadd18f172d52391a062c4eeb265136e0f902c37a0d248234a9a58def9
contracts/saga/helpers/IntervalIteratorMockup.sol	0c43f027f3d70a352b307208237552742d81443177266325c4e984149038a261
contracts/saga/helpers/MintListenerMockup.sol	9bd3ed5406605f3bcc9cd4c45b7695abd9f1e7f50ad7fd26d0343df115c46568
contracts/saga/helpers/MintManagerExposure.sol	1e55f9c4adc60f360382038dfcfaada57711e6cb386e9385381c5419306c4709
contracts/saga/helpers/PriceBandManagerMockup.sol	0c77510e1a5b30d99f2e8c5a0f2225b5ea15b94df3bacf4557e0e899689ede0a
contracts/saga/helpers/PriceCalculatorExposure.sol	7fd0b33017cbbc42e8ff88e148b0b5c56527794d0092bf26fbe4b5c9d658260a
contracts/saga/helpers/PriceCalculatorMockup.sol	f3c84531d54d8bf0698c20e5cdc1ea2d1f764a9020d51cf1a3c4fde3fe4f00e8
contracts/saga/helpers/RedButtonMockup.sol	5eb41b89b018e696d86cbb58d0422fa5053e11cd1f3790227ccd99c74d54b265
contracts/saga/helpers/ReserveManagerMockup.sol	77c063efa57369bc3610f24c606bedca7c21afb8d7c33b883531b245cd45c945
contracts/saga/helpers/SagaModelMockup.sol	84c0c346e96a5230ed7b7adf2bf259f0e092627e2a1dcf4029df876eb8679afd
contracts/saga/helpers/SagaModelStateMockup.sol	4119ee929b1ce1e8ca9f81f0675244738376dfca990b1773fd4b0732c2c0fa8c
contracts/saga/helpers/SGAAuthorizationManagerMockup.sol	972075253ac5c5a04f826c4eda28ed0131c9e09a47c45f3eb8b2903e197c7b5f
contracts/saga/helpers/SGATokenManagerMockup.sol	81481b7648a1aa96a76d2e35532e782107174f214ddc46301c0d39f4ad6eca25
contracts/saga/helpers/TimeManagerExposure.sol	ff819f39c466e6ad99dadb5dd48e842522c5069fb97e6101ebf3a926ba3369bd
contracts/saga/helpers/TimeManagerMockup.sol	4d4e48204c807ec36908dda65cfd8c78920a293581f09e0c29ecee8366064962
contracts/saga/helpers/TradingConverterMockup.sol	0d7ff192654e23c9ac41ed7854a7c12f8d97d05c52c915bea76b03b5aec75609
contracts/saga/helpers/TradingDataSourceMockup.sol	9e2f732f340faff79189d02e7a29f3bddf73918b8192662dafe63cfef0f9f456
contracts/saga/helpers/TradingManagerMockup.sol	00b0d9864563ef9f8c3141c8349aa29d2a26fceac87d1cd57d7cb4336728ee50
contracts/saga/helpers/TransactionConverterMockup.sol	27698940a6a41b232b2e7235aef79a6efc45f5c30af5f358d6f2c41f1b969365
contracts/saga/helpers/TransactionLimiterMockup.sol	3a920e996aa231ed1e1470d86e225c412b135d7939361be90a79f8b1d71b2727
contracts/saga/helpers/TransactionManagerMockup.sol	3fd8ed8844707296000e3032b4b4aed7a099ced65dd194e7aa28b04d2aae06d2
contracts/saga/InterestConverter.sol	79c221b310c1e173138c28c9773a07a4a2217b4a90ac0c11d9d9eac95c75ccbd
contracts/saga/interfaces/IDataSource.sol	9891c0e9831c093b583f9f086f6f31f2f33b7a091efe967db00a74ec27088411
contracts/saga/interfaces/IDebtHandler.sol	2664b6b12b8bba7c329a8906b1449b9992813d0d3f4a8e38fb2b57efa411fd24

contracts/saga/interfaces/IDebtManager.sol	4d398d85cbbcba1ddc1fcae3942eed4f8c1dbb555a7edcc559f224c60a2a2074
contracts/saga/interfaces/IDebtQueue.sol	1ea07ba983b63bab264209028af765e9c0f02f65363dfa6086fcd8cb3e68792a
contracts/saga/interfaces/IInterestConverter.sol	e689a511063a3a586921696f2500212926b2b6de46a79069dc219b7b77bc3076
contracts/saga/interfaces/IIntervalIterator.sol	c44e37e193bc5fc9b2624bf1ca98427fdfbfe88de8f9f2a0b250fa0f3a080a8a
contracts/saga/interfaces/IMintListener.sol	8d6f9e24031a652cb30f5651c01f4e68465e7bd89957450f889334d9e6b4f143
contracts/saga/interfaces/IPriceBandManager.sol	094ddd627e73d3cb041b5764cb11a1b3900508c02b0bdcb8d3559d723e27dbe2
contracts/saga/interfaces/IPriceCalculator.sol	699d405b757e75f90371e0d5f7cae5f4091dff7a92052e6f0257a4e584c260eb
contracts/saga/interfaces/IRedButton.sol	7b7a80ed641178bb88e70b1a128a87b62695d64e7e2cfb1bc0a0494c3cc530ae
contracts/saga/interfaces/IReserveManager.sol	59ba0043082daad651a9f65cb0feed329914e8833c92a54b4e6306ac66f81b1c
contracts/saga/interfaces/ISagaModel.sol	49805d8346e74e9cfa6e9accb32b7501e0a30b60d859780d7c983fb1094f9e81
contracts/saga/interfaces/ISagaModelState.sol	1e7b63612b37569fe9442e9e67b417d8f73d36a2a245eba41025d67c39a6126c
contracts/saga/interfaces/ISGAAuthorizationManager.sol	78899c5626bc4cd62f6c7fb4ba265c259f1159d9f3e8af3f39257c23509d50c1
contracts/saga/interfaces/ISGATokenManager.sol	e833e3ec34fe91e40a89f88629502fe4ff797d40aef3115c963b2e1945986720
contracts/saga/interfaces/ITimeManager.sol	21018d4e13d47760164202d0ff4d2d781c4a40a97384efa2d36130586972f4b4
contracts/saga/interfaces/ITradingConverter.sol	d3355ddae2925c30bdbe49e0d290bd367225db8646563112a046e0d88c4d4aa8
contracts/saga/interfaces/ITradingDataSource.sol	4935abd2f47b8678cf10d4d4e9a45361e06a19630d84bb59a38b04120e2ed86e
contracts/saga/interfaces/ITradingManager.sol	77f2c8aabb798fed1df0a492fefc4bcf6812ff9dd3974f2ed57c3ce5fb81ab15
contracts/saga/interfaces/ITransactionConverter.sol	3069b47aa4f92529d70d5fc4242432ad7df49f1eb8db849963bf75cc8cafad4e
contracts/saga/interfaces/ITransactionLimiter.sol	5d26c77d4143ca2ebb0476ee3970feae4be49228ae6520866ef03b48cb61330f
contracts/saga/interfaces/ITransactionManager.sol	21b5ced7479887836f16282cd4d71e79954f7a93170c40d3c6c068da52384cb4
contracts/saga/IntervalIterator.sol	2c53f09d4968be50162b6518d20b6c40f04026c6fd0384c5f8a603586ce467e7
contracts/saga/MintManager.sol	866a84c6444b9e333eca3e0b84da1ac8eaff1bdef8630795b52f18014a6b6c9a
contracts/saga/PriceBandManager.sol	5ec8938bef63424f4feb1051e37454ebf0b69255fea34bad4d1cad8d6e139383
contracts/saga/PriceCalculator.sol	00cd461a28c2259340952eddb4e57c454bb87dab719c06b70196289d35714b82
contracts/saga/RedButton.sol	ef5b5bce40019e9a8f9cd99c875f6828fc36b0c5c45a5173a811252ed69f09b9
contracts/saga/ReserveManager.sol	d679417bafa769f9a852f47627af15e83767609e7e8724483b5413288250113e
contracts/saga/SagaModel.sol	86715aa420b5d740fe6fbfa9b0e66d2aa3d1125e470a4777510cde4ec13f951a
contracts/saga/SagaModelState.sol	9a173d4850af436da45664b253a563b96ec80b3a1278ea3bb1c084be28dda1b8
contracts/saga/SGAAuthorizationManager.sol	5870d949ef3659ce3ae6c4ba4a4037b7ff3482062f1a35e034fae2ca35be8694
contracts/saga/SGAToken.sol	93c3408d9a4188393ae8491ec9a66e182b9911d31ad605cd4ad29344ebcc382d
contracts/saga/SGATokenManager.sol	d2a069bc2db58bf7fdb736d01446cba6b164fdfe000bb62d239f724a28a81eff
contracts/saga/TimeManager.sol	4f44e7c1c28a3bf623449af113a63eced63afcfb05a44c905681b736a0bc8564
contracts/saga/TradingConverter.sol	d40497522dba1d7c0553c16dd73915bc11682a52d105f9146bf25fee450343cc
contracts/saga/TradingDataSource.sol	7329e6d81a8bc8986cc26816c732bb3e1dcdb4219260274bf20584ef570acdba
contracts/saga/TradingManager.sol	9802f30fa801871f85d2f730ef9f8273978e269558323615d18fd72a9fe52423
contracts/saga/TransactionConverter.sol	133689207c30bdf1dc1e71dd174ccf025f091779674da7fdff002316d70721f1
contracts/saga/TransactionLimiter.sol	bdb3fa7d3592e7bc2804aa85d31edbca970a0786d67a1c8489d82c6ccb9076ab
contracts/saga/TransactionManager.sol	f4a81342489027fd6f7f7f8c0c697ace57bbfd067999c83dcdaba07a4238956d
contracts/utils/Adminable.sol	f82aac08dba01613efd4990ea917e6978263d19fbef5f2c3641deee3a72808f0
contracts/utils/helpers/MultiSigWalletExposure.sol	f0ea1436c049043fbb54492e43e9649729fc354253e28e688218cb576edd1b04
contracts/utils/helpers/MultiSigWalletTestCalls.sol	d4219f873ee034af4db5cb74d4945daf44ebcdd599050917c8fe30c65bdf0c23

contracts/utils/helpers/MultiSigWalletTestToken.sol	7106e4d474a7d2ca69cd5eb6259fd71d99750403c0f98c230f37921e2cd79da0
contracts/utils/MultiSigWallet.sol	290cae6e3e31a8adc6447b45daa930ce2ad368e5d7d0eaf5cf248b822ce54ead
migrations/1_initial_migration.js	533b7003b8ea3b5249aa48706cb4078fc6c04ee4936436f17a550f36cd979ae3
migrations/2_deploy_contracts_phase1.js	15f0061224b8e8b4cedb5b0c7617a2a97f80817c71bc664f66f6e4ee980caefd
migrations/3_deploy_contracts_phase2.js	d03d77b467fb25fcf23164a4816e2fdf75e76613592022eec49507da076c7576
package.json	03bfe1937cae64e0b4839035def541100a74e8611fa3f1937f9c243692fbc15d
scripts/fix-modules.js	184f99f31a8c6a4bfb6e9f46ef3c968139e71f96bde897878c050c6c35cbe4dc
scripts/rebuild-all.js	01bd44bae4e7a4f16e54de96a30f3cd40b2b63dfd9f2f31daca1236a3f9d8c9f
scripts/run-tests.js	23fd7e9c7f8140a58e52ceb6b4a470fc400ca6ed021d6ce156339ac665467742
test/authorization/AuthorizationDataSourceUnitTest.js	14d50eafdcaf7141839d79887e75d73b09215d2b49774bcd51a8794a6905f71e
test/exceptions.js	89eff46ad85c873ec1115790fb95933fe25ba52f5a6b2a53f73c1c9fc6eb866e
test/registry/ContractAddressLocatorHolderUnitTest.js	a11e60417e47a99be5fba30259702090f14c2427d833335a4e41bf9a58ce82e4
test/registry/ContractAddressLocatorProxyUnitTest.js	4639d90b0205f4b822713117d94e205903435981b51b56362915178cd3030245
test/registry/ContractAddressLocatorUnitTest.js	11b18046d529383cdab50b82e0c4d0e5abd0fa03900fad189bcfb36607cf819a
test/saga-genesis/ConversionManagerUnitTest.js	a9c75bada659461b347a7474f181b521145f6939473f2fbb5279e3bddfc204dc
test/saga-genesis/helpers/ConversionManager.js	f4f681a8cef94a757e5ea7acbfa00e9e620c2f77f7edeaf586d9fdd7a99aefe9
test/saga-genesis/SGNAuthorizationManagerUnitTest.js	fd28eae1db9cccfe9fdae47a60354f21ee40b1ac1b30e300e137084db4feae38
test/saga-genesis/SGNTokenFuncTest.js	52affdb97e4c67af4334b8e8af0594cf485cefab19c53961beedd1fa9eb3421e
test/saga-genesis/SGNTokenManagerUnitTest.js	a3474fdced3afb7ae3666274bd111a4e6c636c17bedd2e157084c7a93282faf8
test/saga-genesis/SGNTokenUnitTest.js	d9ba5d6d8ea9ab9e3f5f1661855202f13324bab3edd0dda109cda74ebb448ab6
test/saga/DataSourceUnitTest.js	5ca9c6f8c70dab3b116210282ebcf6a096fea98c63ecbacd8cc2c67e35d0f830
test/saga/DebtManagerUnitTest.js	83763c45aa9e5568f73f94012a89e965879da8094312d8a1fdd1ce44ef44c500
test/saga/DebtQueueUnitTest.js	41d7dc177aa63ecbddc06dc9bccfc5f963e96963e396dccd163bc05164fb0b15
test/saga/helpers/DataSource.js	cfb1b32412c58a49c6a55c57aa775c8931a8b312fa77bc4e5119b2a19e8c105d
test/saga/helpers/sequences/BuyAll.js	7b81659b71adec346e8fcd56d0aacd71549252373c11ef8584e639263570ef1e
test/saga/helpers/sequences/BuyAllSellAll.js	3df32cc098ad7f49bd889da628923c026e26e85250de029ca36c49a31c8b7945
test/saga/helpers/sequences/BuySomeSellSome.js	03b4ad26722f230da2171ab85f69719d9a01c57a90f603630a1df7c7fe3be8d5
test/saga/InterestConverterUnitTest.js	85de9d82ef51725af2f8328fbf002301740c6a05e83c1a828fe7acb5ecef83b0
test/saga/IntervalIteratorUnitTest.js	399f6aecbdc1189f5e91b9b9e858ca04ffebff53bbfddf236d2e077c833334bc
test/saga/MintManagerUnitTest.js	4c09e0a252424856bbbb10619e3ac052acbe9da2cf6347b96acf026e9640ee1a
test/saga/PriceBandManagerUnitTest.js	ec61f5fdf801ab461c05b5c9695a1269b58d1f63a9559bb0ccd8d8ac301decf1
test/saga/PriceCalculatorExtUnitTest.js	90c512c8d4419ee3d75041e96bed4c6d7776e0285ba05bdbc522b0d8a36496bc
test/saga/PriceCalculatorIntUnitTest.js	6c64d99f3e85477d303464f12b13a52b53d39a37a725a962cb65ca184f87011d
test/saga/RedButtonUnitTest.js	bc9aa86c19c1f345ee2aa18d104fd765cf9342b91d638dd925fb07c8f46bea15
test/saga/ReserveManagerUnitTest.js	11ad2fbbd2d15d6658d622fa075525c0fd89e5829ad22e01ba24d7fdacb2040e
test/saga/SagaModelFuncTest.js	439cd3f742a7c6af42993365eeed74863ea24a2f05f0c1d95e5a7ce5cdba72b8
test/saga/SagaModelStateUnitTest.js	b9cb91195450a4e23c0a36806d4c7d4d9189ac7301246d26d224189f23128dd3
test/saga/SagaModelUnitTest.js	4d767b68ef1293a855434509e87d9b3074db7809b05eed5e3f9e63a015ca6fd9
test/saga/SGAAuthorizationManagerUnitTest.js	32d788fba878d1c22b16de0356a7da5c8a9b9fc384542ab67b0b96228399d0ee
test/saga/SGATokenFuncTest.js	af55756a91979e6d70b98ee50e72a4cc585dde9a88e1d7e5ab1728002e55e19b
test/saga/SGATokenManagerUnitTest.js	974a7a58341be8e43888bb3622414ca9ae5ce685fa385b8781a63294a5103bb7

test/saga/SGATokenUnitTest.js	f374cceb3aae1bcc3a65f0b8d1cc4f26569b301fb2b5bd8b32b66c347f3a56ed
test/saga/SystemTest.js	d47ff93c4913ee1c1f457607d2c336fdf4853b4b30dedb2d552462dc74f890c5
test/saga/TimeManagerUnitTest.js	259f93dfdfb03f73ccc4d7a4104d44fa4dab8f4e61231fdec814899560bcb7cd
test/saga/TradingConverterUnitTest.js	e252ded5b4a96e959c22f8b987a51a4ae5a1113fdbf8c545ff95748eb168276a
test/saga/TradingDataSourceUnitTest.js	34e82c7cdc3f3955ee8e79668a3d5013ab97ac3051fdca3215141380c76ab878
test/saga/TradingManagerUnitTest.js	5b25f0943fb5dd109e8a0f7f3a5ecd9e549178eccb7adc47ffdd8abb2639a800
test/saga/TransactionConverterUnitTest.js	84cb0fda45e445677d807257f01c5c5d2131b1896a0e6df9813f65ba473facfa
test/saga/TransactionLimiterUnitTest.js	00d0558fbca1c42df4042fc707b2e46ec3967b2f0821cb2a96ae5f8f24286e7c
test/saga/TransactionManagerUnitTest.js	3bcd95740702ef6b4a58f9a28ae1d2f9d9a830b888894097e730ab5a7541c8f2
test/utils/AdminableUnitTest.js	1f4f4e08a504d61845984e995ed07e964b742c24208a1a0c202e1e6c5ef03bda
test/utils/helpers/utilsMultiSig.js	de83e0755e229ab0a65242629069d3f206cf7c222400ae2556a885a649e89176
test/utils/MultiSigWalletTestExecutionAfterRequirementsChanged.js	d3d896c3312bf9238db16a99894bfc30a6206a63646d53511b6404cbaa719054
test/utils/MultiSigWalletTestExternalCalls.js	35d824070bd38cb446698387e0ff780b314b53ca284beb521e62cfb884d44ac5
test/utils/MultiSigWalletTestFallback.js	838b29fd939f3e8eadd403b1647806042787e295e3d84b6a1f23ffe6c58ee301
test/utils/MultiSigWalletTestOwnership.js	cecd87cdac7e06a7d5fb8b6e296c90cd483fe3fdf2c1d52bf414826e8b8fcd1f
test/utils/MultiSigWalletTestRequirements.js	6aa2d2947787e83946636f7ad94832843c3c7e589e64f1ec1edd559dc2afb21a
test/utils/MultiSigWalletTestSequence.js	7855bf372dfcecdcbc945d3078f2ba75dd11ea0bf3f8c3298805c16dc4dd6b05
truffle-config.js	81581fd3ed56d736807cc3d7795d35f39b185e517794686d89cafa8c40adb4d6
.solcover.js	0709a97fcef6f79f811d8a5634508920ade6642e16ce4208c0318113c31cb722

Appendix B: MultiSigWallet Analysis

We have verified that the **contracts/utils/MultiSigWallet.sol** is compliant the original Gnosis MultiSigWallet v1.6.0 and was adapted to Solidity v.0.4.25 by the following modifications:

1. Added the **public** modifier to the **fallback** function.
2. Emitting events using the **emit** keyword.
3. Redefined the constructor using the **constructor** keyword.
4. Read-only methods are defined as **view** (instead of as **constant**).
5. Added missing return from the **isConfirmed** function.

The provided implementation differs from the original by:

- It's not possible to replace an owner with a 0x0 address. This is the recommended behavior and a proper fix by the Saga Core team.

We would also recommend changing the fallback function visibility to **external** and specify the data location of the **address[], uint[],** and **bytes** input and return variables.

Consider sending a pull request to the official repository so that these changes are integrated back into the main development.

MultiSigWallet.sol is only partially covered by tests **(98.82%** statements, **93.18%** branches, **100%** functions, **99.01%** lines), which is the de-facto standard Gnosis MultiSigWallet smart contract[6].

[6] We'd encourage the team to bring MultiSigWallet's code coverage to a perfect 100% as well and will be more than happy to provide the team with our own test suite, which already achieves that.

Appendix C: Test Cases

Suite Name	Passed	Failed
AdminableUnitTest data-structure management	2	0
AdminableUnitTest functionality assertion	2	0
AdminableUnitTest security assertion	6	0
AuthorizationDataSourceUnitTest functionality assertion	10	0
AuthorizationDataSourceUnitTest security assertion	8	0
ContractAddressLocatorHolderUnitTest function isSenderAddressRelates	2	0
ContractAddressLocatorHolderUnitTest functionality assertion	3	0
ContractAddressLocatorHolderUnitTest security assertion	1	0
ContractAddressLocatorProxyUnitTest function upgrade	1	0
ContractAddressLocatorProxyUnitTest functionality assertion	3	0
ContractAddressLocatorProxyUnitTest security assertion	2	0
ContractAddressLocatorUnitTest function isContractAddressRelates	6	0
ContractAddressLocatorUnitTest functionality assertion	3	0
ContractAddressLocatorUnitTest security assertion	3	0
ETHConverterUnitTest function fromEthAmount	100	0
ETHConverterUnitTest function toEthAmount	100	0
ETHConverterUnitTest function toSdrAmount	100	0
ETHConverterUnitTest functionality assertion	7	0
ETHConverterUnitTest security assertion	1	0
IntervalIteratorUnitTest function grow with running = false	100	0
IntervalIteratorUnitTest function grow with running = true	100	0
IntervalIteratorUnitTest function shrink with running = false	100	0
IntervalIteratorUnitTest function shrink with running = true	90	0
IntervalIteratorUnitTest functionality assertion	2	0
IntervalIteratorUnitTest integrity assertion	1	0
IntervalIteratorUnitTest security assertion	2	0
MintingPointTimersManagerUnitTest emulated assertion	3	0
MintingPointTimersManagerUnitTest integrity assertion	2	0
MintingPointTimersManagerUnitTest security assertion	2	0
MintingPointTimersManagerUnitTest simulated assertion	100	0
MintManagerUnitTest authorization assertion	2	0
MintManagerUnitTest functionality assertion	20	0

ModelCalculatorExtUnitTest accuracy assertion	17840	0
ModelCalculatorIntUnitTest functionality assertion	500	0
ModelDataSourceUnitTest functionality assertion	4	0
ModelDataSourceUnitTest integrity assertion	1	0
ModelDataSourceUnitTest security assertion	2	0
MonetaryModelFuncTest functional assertion: file BuyAll.js	100	0
MonetaryModelFuncTest functional assertion: file BuyAllSellAll.js	3	0
MonetaryModelFuncTest functional assertion: file BuySomeSellSome.js	4000	0
MonetaryModelStateUnitTest functionality assertion	2	0
MonetaryModelStateUnitTest security assertion	2	0
MonetaryModelUnitTest function buy	66	0
MonetaryModelUnitTest function sell	66	0
MonetaryModelUnitTest security assertion	2	0
MultiSigWalle	6	0
MultiSigWallet fallback function	2	0
MultiSigWallet function revokeConfirmation	4	0
MultiSigWallet functional assertion	8	0
MultiSigWallet security assertion	16	0
PaymentManagerUnitTest combination	2	0
PaymentManagerUnitTest combination	2	0
PaymentManagerUnitTest combination 1	3	0
PaymentManagerUnitTest combination 1	4	0
PaymentManagerUnitTest combination 1	4	0
PaymentManagerUnitTest combination 1	4	0
PaymentManagerUnitTest combination 1	5	0
PaymentManagerUnitTest combination	2	0
PaymentManagerUnitTest combination	3	0
PaymentManagerUnitTest combination	3	0
PaymentManagerUnitTest combination	3	0
PaymentManagerUnitTest combination	3	0
PaymentManagerUnitTest combination	3	0
PaymentManagerUnitTest combination	3	0
PaymentManagerUnitTest combination	3	0
PaymentManagerUnitTest function computeDifferPayment when the payment-queue is empty	25	0
PaymentManagerUnitTest function computeDifferPayment when the payment-queue is not empty	25	0

PaymentManagerUnitTest function getNumOfPayments	1	0
PaymentManagerUnitTest function getPaymentsSum	1	0
PaymentManagerUnitTest function setMaxNumOfPaymentsLimit	2	0
PaymentManagerUnitTest function settlePayments with maximum 1 payment to handle	3	0
PaymentManagerUnitTest function settlePayments	8	0
PaymentManagerUnitTest security assertion	1	0
PaymentQueueUnitTest function addPayment	8	0
PaymentQueueUnitTest function clean	5	0
PaymentQueueUnitTest function getNumOfPayments	6	0
PaymentQueueUnitTest function getPayment	7	0
PaymentQueueUnitTest function getPaymentsSum	6	0
PaymentQueueUnitTest function removePayment	8	0
PaymentQueueUnitTest function updatePayment	7	0
PriceBandCalculatorExtUnitTest accuracy assertion	17840	0
PriceBandCalculatorIntUnitTest integrity assertion	6	0
RateApproverUnitTest constructor	2	0
RateApproverUnitTest function approveRate	6	0
RateApproverUnitTest function setRateBounds	5	0
ReconciliationAdjusterUnitTest function adjustBuy	100	0
ReconciliationAdjusterUnitTest function adjustSell	100	0
ReconciliationAdjusterUnitTest functionality assertion	4	0
ReconciliationAdjusterUnitTest security assertion	1	0
RedButtonUnitTest functionality assertion	2	0
RedButtonUnitTest security assertion	2	0
ReserveManagerUnitTest function setThresholds	2	0
ReserveManagerUnitTest function setWallets	2	0
ReserveManagerUnitTest functionality assertion	23	0
ReserveManagerUnitTest security assertion	6	0
SGAAuthorizationManagerUnitTest function isAuthorizedForPublicOperation	2	0
SGAAuthorizationManagerUnitTest functionality assertion	16	0
SGATokenFuncTest functional assertion: file BuyAll.js	100	0
SGATokenFuncTest functional assertion: file BuyAllSellAll.js	3	0
SGATokenFuncTest functional assertion: file BuySomeSellSome.js	4000	0
SGATokenManagerUnitTest authorization assertion	5	0
SGATokenManagerUnitTest functionality assertion	14	0
SGATokenManagerUnitTest payment-handling assertion	1	0

SGATokenManagerUnitTest red-button assertion	3	0
SGATokenManagerUnitTest reserve-amount assertion	2	0
SGATokenManagerUnitTest reserve-wallet assertion	2	0
SGATokenManagerUnitTest security assertion	11	0
SGATokenManagerUnitTest token-selling assertion	3	0
SGATokenUnitTest explicit-payment assertion	3	0
SGATokenUnitTest functionality assertion	9	0
SGATokenUnitTest implicit-payment assertion	3	0
SGATokenUnitTest information-retrieval assertion	3	0
SGATokenUnitTest security assertion	3	0
SGATokenUnitTest token-selling assertion	2	0
SGAWalletsTradingLimiterUnitTest function getLimiterValue	1	0
SGAWalletsTradingLimiterUnitTest function getUpdateWalletPermittedContractLocatorIdentifier	1	0
SGAWalletsTradingLimiterUnitTest function updateWallet	3	0
SGNAuthorizationManagerUnitTest functionality assertion	14	0
SGNConversionManagerUnitTest illegal amount, legal index	105	0
SGNConversionManagerUnitTest legal amount, illegal index	1	0
SGNConversionManagerUnitTest legal amount, legal index	105	0
SGNTokenFuncTest function convert	105	0
SGNTokenFuncTest function transfer when selling	3	0
SGNTokenFuncTest function transfer	6	0
SGNTokenFuncTest function transferFrom	6	0
SGNTokenFuncTest token-selling	103	0
SGNTokenManagerUnitTest authorization assertion	3	0
SGNTokenManagerUnitTest conversion assertion	2	0
SGNTokenManagerUnitTest functionality assertion	6	0
SGNTokenManagerUnitTest security assertion	4	0
SGNTokenManagerUnitTest token-selling assertion	1	0
SGNTokenUnitTest functionality assertion	4	0
SGNTokenUnitTest security assertion	1	0
SGNTokenUnitTest token-minting assertion	3	0
SGNTokenUnitTest token-selling assertion	3	0
SGNWalletsTradingLimiterUnitTest function calcSGNMinimumLimiterValue	101	0
SGNWalletsTradingLimiterUnitTest function getLimiterValue	3	0
SGNWalletsTradingLimiterUnitTest function setSGNMinimumLimiterValue	7	0

SGNWalletsTradingLimiterUnitTest function updateWallet	3	0
ShareHolderTest functional assertion	100	0
SystemTest functional assertion: file BuyAll.js	101	0
SystemTest functional assertion: file BuyAllSellAll.js	4	0
SystemTest functional assertion: file BuySomeSellSome.js	4001	0
TradingClassesUnitTest data-structure management	2	0
TradingClassesUnitTest functionality assertion	24	0
TradingClassesUnitTest security assertion	1	0
TransactionLimiterUnitTest function setMaxDiff	2	0
TransactionLimiterUnitTest functionality assertion	6	0
TransactionLimiterUnitTest security assertion	4	0
TransactionManagerUnitTest functionality assertion	2	0
TransactionManagerUnitTest security assertion	2	0
WalletsTradingDataSourceUnitTest functionality assertion	4	0
WalletsTradingDataSourceUnitTest modifier onlyWalletsTradingLimiters	2	0
WalletsTradingDataSourceUnitTest security assertion	2	0
WalletsTradingLimiterValueConverterUnitTest function toLimiterValue	100	0
WalletsTradingLimiterValueConverterUnitTest functionality assertion	3	0
WalletsTradingLimiterValueConverterUnitTest security assertion	1	0

Appendix D: Natural Logarithm Tests

Described below is the precision testing of the natural logarithm approximation using Taylor series with fixed-point operations, as it was implemented in **contracts/saga/PriceCalculator.sol**:

Input	Control	PriceCalculator.sol	Change (%)
1	0.000000000000000	0.000000000000000	0.0000000000000%
1.19	0.173953307123438	0.173953307123438	0.0000000000000%
1.38	0.322083499169113	0.322083499169113	-0.0000000000001%
1.57	0.451075619360217	0.451075619360216	-0.0000000000002%
1.76	0.565313809050061	0.565313809050060	-0.0000000000001%
1.95	0.667829372575655	0.667829372575655	-0.0000000000001%
2.14	0.760805829033760	0.760805829033760	0.0000000000000%
2.33	0.845868267577609	0.845868267577609	0.0000000000000%
2.52	0.924258901523332	0.924258901523331	-0.0000000000001%
2.71	0.996948634891610	0.996948634891609	0.0000000000000%
2.9	1.064710736992430	1.064710736992420	-0.0000000000008%
3.09	1.128171090909650	1.128171090909650	-0.0000000000004%
3.28	1.187843422396050	1.187843422396050	-0.0000000000002%
3.47	1.244154593958770	1.244154593958760	-0.0000000000006%
3.66	1.297463147413280	1.297463147413270	-0.0000000000004%
3.85	1.348073148299690	1.348073148299690	-0.0000000000002%

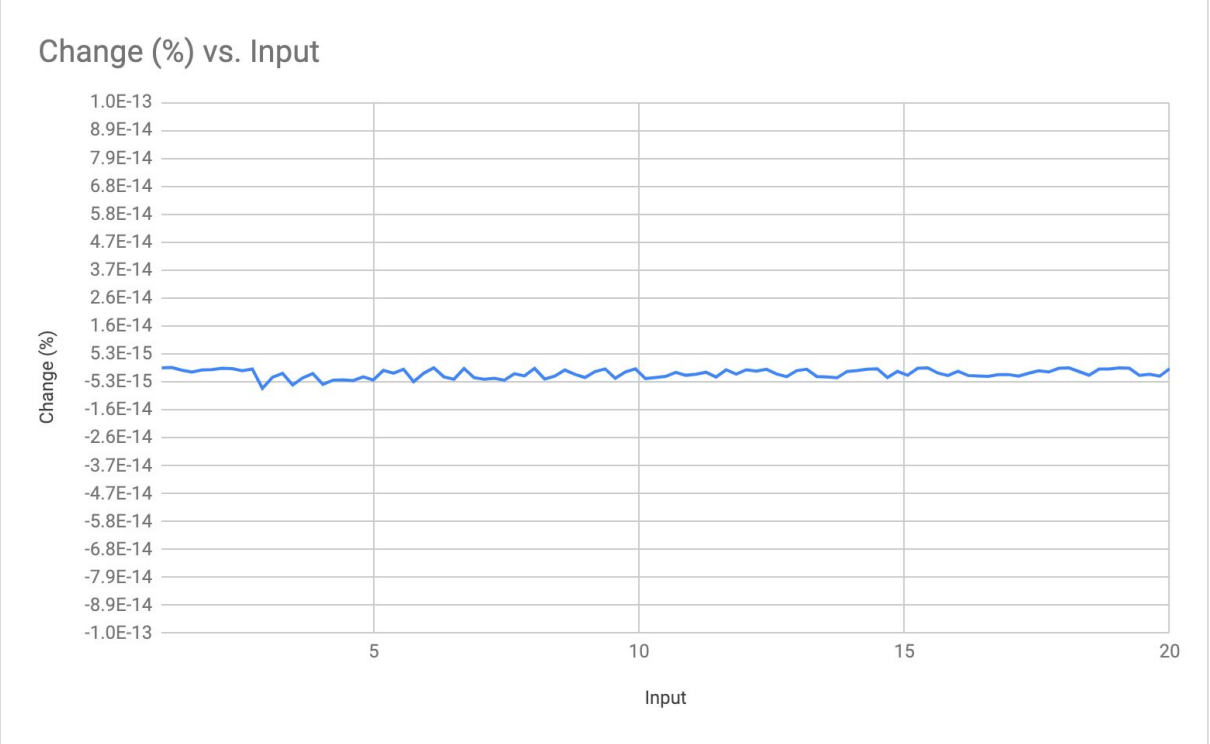
4.04	1.396244691973060	1.396244691973050	-0.0000000000006%
4.23	1.442201993058190	1.442201993058180	-0.0000000000005%
4.42	1.486139696089610	1.486139696089600	-0.0000000000004%
4.61	1.528227857008560	1.528227857008550	-0.0000000000005%
4.8	1.568615917913850	1.568615917913840	-0.0000000000003%
4.99	1.607435909763430	1.607435909763420	-0.0000000000005%
5.18	1.644805056271390	1.644805056271390	-0.0000000000001%
5.37	1.680827908520770	1.680827908520770	-0.0000000000002%
5.56	1.715598108262490	1.715598108262490	-0.0000000000001%
5.75	1.749199854809260	1.749199854809250	-0.0000000000005%
5.94	1.781709133374550	1.781709133374550	-0.0000000000002%
6.13	1.813194749948120	1.813194749948120	0.0000000000000%
6.32	1.843719208158770	1.843719208158760	-0.0000000000003%
6.51	1.873339456220480	1.873339456220470	-0.0000000000004%
6.7	1.902107526396920	1.902107526396920	0.0000000000000%
6.89	1.930071085025570	1.930071085025560	-0.0000000000004%
7.08	1.957273907705630	1.957273907705620	-0.0000000000004%
7.27	1.983756291545430	1.983756291545420	-0.0000000000004%
7.46	2.009555414215670	2.009555414215660	-0.0000000000005%
7.65	2.034705647838440	2.034705647838440	-0.0000000000002%
7.84	2.059238834362320	2.059238834362310	-0.0000000000003%

8.03	2.083184527958670	2.083184527958670	0.0000000000000%
8.22	2.106570209068090	2.106570209068080	-0.0000000000004%
8.41	2.129421473984860	2.129421473984850	-0.0000000000003%
8.6	2.151762203259460	2.151762203259460	-0.0000000000001%
8.79	2.173614711697090	2.173614711697080	-0.0000000000002%
8.98	2.194999882314110	2.194999882314100	-0.0000000000004%
9.17	2.215937286268370	2.215937286268370	-0.0000000000001%
9.36	2.236445290489500	2.236445290489500	0.0000000000000%
9.55	2.256541154492640	2.256541154492630	-0.0000000000004%
9.74	2.276241117654440	2.276241117654440	-0.0000000000002%
9.93	2.295560478057080	2.295560478057080	0.0000000000000%
10.12	2.314513663859320	2.314513663859310	-0.0000000000004%
10.31	2.333114298028870	2.333114298028860	-0.0000000000004%
10.5	2.351375257163480	2.351375257163470	-0.0000000000003%
10.69	2.369308725036950	2.369308725036950	-0.0000000000002%
10.88	2.386926241427800	2.386926241427790	-0.0000000000003%
11.07	2.404238746720550	2.404238746720540	-0.0000000000002%
11.26	2.421256622711540	2.421256622711540	-0.0000000000002%
11.45	2.437989730000250	2.437989730000240	-0.0000000000003%
11.64	2.454447442303290	2.454447442303290	-0.0000000000001%
11.83	2.470638677990300	2.470638677990290	-0.0000000000002%

12.02	2.486571929107060	2.486571929107060	-0.0000000000001%
12.21	2.502255288122610	2.502255288122610	-0.0000000000001%
12.4	2.517696472610990	2.517696472610990	-0.0000000000001%
12.59	2.532902848056260	2.532902848056250	-0.0000000000002%
12.78	2.547881448949390	2.547881448949380	-0.0000000000003%
12.97	2.562638998328350	2.562638998328350	-0.0000000000001%
13.16	2.577181925897170	2.577181925897170	-0.0000000000001%
13.35	2.591516384846260	2.591516384846250	-0.0000000000003%
13.54	2.605648267484130	2.605648267484120	-0.0000000000003%
13.73	2.619583219779880	2.619583219779870	-0.0000000000004%
13.92	2.633326654906270	2.633326654906270	-0.0000000000001%
14.11	2.646883765864720	2.646883765864720	-0.0000000000001%
14.3	2.660259537265860	2.660259537265860	-0.0000000000001%
14.49	2.673458756332590	2.673458756332590	0.0000000000000%
14.68	2.686486023186370	2.686486023186360	-0.0000000000004%
14.87	2.699345760472060	2.699345760472060	-0.0000000000001%
15.06	2.712042222371750	2.712042222371740	-0.0000000000003%
15.25	2.724579503053420	2.724579503053420	0.0000000000000%
15.44	2.736961544596630	2.736961544596630	0.0000000000000%
15.63	2.749192144433390	2.749192144433380	-0.0000000000002%
15.82	2.761274962339510	2.761274962339500	-0.0000000000003%

16.01	2.773213527008620	2.773213527008620	-0.0000000000001%
16.2	2.785011242238340	2.785011242238330	-0.0000000000003%
16.39	2.796671392755740	2.796671392755730	-0.0000000000003%
16.58	2.808197149707150	2.808197149707140	-0.0000000000003%
16.77	2.819591575835120	2.819591575835110	-0.0000000000003%
16.96	2.830857630363760	2.830857630363750	-0.0000000000003%
17.15	2.841998173611950	2.841998173611940	-0.0000000000003%
17.34	2.853015971352400	2.853015971352390	-0.0000000000002%
17.53	2.863913698933140	2.863913698933140	-0.0000000000001%
17.72	2.874693945176930	2.874693945176930	-0.0000000000002%
17.91	2.885359216072620	2.885359216072620	0.0000000000000%
18.1	2.895911938271780	2.895911938271780	0.0000000000000%
18.29	2.906354462402770	2.906354462402770	-0.0000000000001%
18.48	2.916689066213540	2.916689066213530	-0.0000000000003%
18.67	2.926917957553630	2.926917957553630	0.0000000000000%
18.86	2.937043277205310	2.937043277205310	0.0000000000000%
19.05	2.947067101572710	2.947067101572710	0.0000000000000%
19.24	2.956991445237560	2.956991445237560	0.0000000000000%
19.43	2.966818263389350	2.966818263389340	-0.0000000000003%
19.62	2.976549454137220	2.976549454137210	-0.0000000000002%
19.81	2.986186860710460	2.986186860710450	-0.0000000000003%

20	2.995732273553990	2.995732273553990	0.0000000000000%



Needless to say that the estimation method in **contracts/PriceCalcualtor.sol** is excellent.

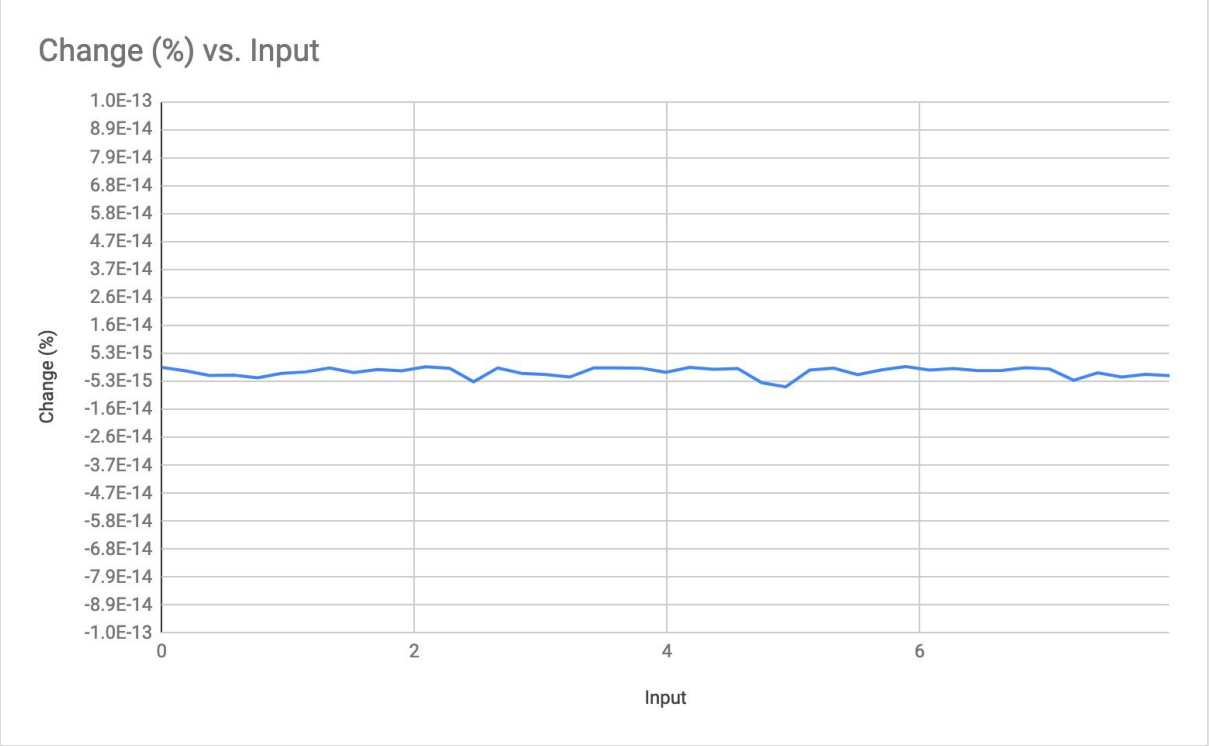
Appendix E: Exponential Function Tests

Described below is the precision testing of the exponential function approximation using Taylor series with fixed-point operations, as it was implemented in **contracts/saga/PriceCalculator.sol**:

Input	Control	PriceCalculator.sol	Change (%)
0	1.000000000000000	1.000000000000000	0.000000000000000
0.19	1.209249597657250	1.209249597657250	-0.000000000000001
0.38	1.462284589434220	1.462284589434220	-0.000000000000003
0.57	1.768267051433740	1.768267051433730	-0.000000000000003
0.76	2.138276220496820	2.138276220496810	-0.000000000000004
0.95	2.585709659315850	2.585709659315840	-0.000000000000002
1.14	3.126768365186160	3.126768365186150	-0.000000000000002
1.33	3.781043387568780	3.781043387568780	0.000000000000000
1.52	4.572225195142160	4.572225195142150	-0.000000000000002
1.71	5.528961477624000	5.528961477624000	-0.000000000000001
1.9	6.685894442279270	6.685894442279260	-0.000000000000001
2.09	8.084915164305060	8.084915164305060	0.000000000000000
2.28	9.776680409528900	9.776680409528900	0.000000000000000
2.47	11.822446851646400	11.822446851646300	-0.000000000000005
2.66	14.296289098677600	14.296289098677600	0.000000000000000
2.85	17.287781840567600	17.287781840567600	-0.000000000000002

3.04	20.905243235092800	20.905243235092700	-0.000000000000003
3.23	25.279656970962900	25.279656970962800	-0.000000000000004
3.42	30.569415021050200	30.569415021050200	0.000000000000000
3.61	36.966052814822500	36.966052814822500	0.000000000000000
3.8	44.701184493300800	44.701184493300800	0.000000000000000
3.99	54.054889363326600	54.054889363326500	-0.000000000000002
4.18	65.365853214009900	65.365853214009900	0.000000000000000
4.37	79.043631699564500	79.043631699564400	-0.000000000000001
4.56	95.583479830066200	95.583479830066200	0.000000000000000
4.75	115.584284527188000	115.584284527187000	-0.000000000000006
4.94	139.770249560003000	139.770249560002000	-0.000000000000007
5.13	169.017118044887000	169.017118044887000	-0.000000000000001
5.32	204.383881992968000	204.383881992968000	0.000000000000000
5.51	247.151127067624000	247.151127067623000	-0.000000000000003
5.7	298.867400967060000	298.867400967060000	-0.000000000000001
5.89	361.405284372286000	361.405284372286000	0.000000000000000
6.08	437.029194718391000	437.029194718391000	-0.000000000000001
6.27	528.477377877687000	528.477377877687000	0.000000000000000
6.46	639.061056569553000	639.061056569552000	-0.000000000000001
6.65	772.784325535150000	772.784325535149000	-0.000000000000001
6.84	934.489134729210000	934.489134729210000	0.000000000000000

7.03	1130.030610186370000	1130.030610186370000	-0.000000000000001
7.22	1366.489060708250000	1366.489060708240000	-0.000000000000005
7.41	1652.426346864480000	1652.426346864480000	-0.000000000000002
7.6	1998.195895104120000	1998.195895104110000	-0.000000000000004
7.79	2416.317582195030000	2416.317582195020000	-0.000000000000003
7.98	2921.931064081480000	2921.931064081470000	-0.000000000000003



Appendix F: SECURITY.md Template[7]

Security Policy

[COMPANY] is committed to resolving security vulnerabilities in our software quickly and carefully. We take the necessary steps to minimize risk, provide timely information, and deliver vulnerability fixes and mitigations required to address security issues.

Reporting a Vulnerability

Security vulnerabilities in [PROJECT] software should be reported by email to [SECURITY_EMAIL]. If you think your report might be eligible for the [PROJECT] Bug Bounty Program, your email should be sent to [BOUNTY_EMAIL].

Your report should include the following:

- Your name (optional).
- Description of the vulnerability.
- Attack scenario or a reproduction.
- Any other details.

Try to include as much information in your report as you can, including a description of the vulnerability, its potential impact, and steps for reproducing it. Be sure to use a descriptive subject line.

You'll receive a response to your email within [X] business days indicating the next steps in handling your report. We encourage finders to use encrypted communication channels to protect the confidentiality of vulnerability reports. You can encrypt your report using our public key. This key is [PUBLIC_PGP_REGISTRY] server and reproduced below.

After the initial reply to your report, our team will endeavor to keep you informed of the progress being made towards a fix. These updates will be sent at least every [X] business days.

Thank you for taking the time to disclose any vulnerabilities you find responsibly.

Responsible Investigation and Reporting

Responsible investigation and reporting include, but isn't limited to, the following:

- Don't violate the privacy of other users, destroy data, etc.

[7] Adapted from https://github.com/paritytech/parity-ethereum/blob/master/SECURITY.md

- Don’t defraud or harm Parity Technologies Ltd or its users during your research; you should make a good faith effort not to interrupt or degrade our services.
- Don't target our physical security measures, or attempt to use social engineering, spam, distributed denial of service (DDOS) attacks, etc.
- Initially report the bug only to us and not to anyone else.
- Give us a reasonable amount of time to fix the bug before disclosing it to anyone else, and give us adequate written warning before disclosing it to anyone else.
- In general, please investigate and report bugs in a way that makes a reasonable, good-faith effort not to be disruptive or harmful to our users or us. Otherwise, your actions might be interpreted as an attack rather than an effort to be helpful.

Bug Bounty Program

Our Bug Bounty Program allows us to recognize and reward members of the [PROJECT] community for helping us find and address significant bugs, in accordance with the terms of the [PROJECT] Bug Bounty Program. A detailed description of eligibility, rewards, legal information, and terms & conditions for contributors can be found on [WEBSITE].

Plaintext PGP Key

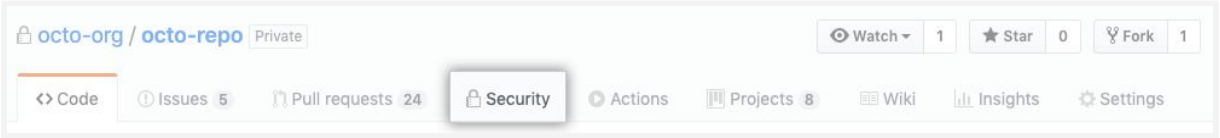
```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQMuBExtIfMRCACe8xHPWBciPfCkdN+4TNUPW04ahDdlSJk5fmzaFcx8GnJILvzt
+0Vbs4HPLUj0yJQz0ZXz+8EPqzs/sqBYZjh5doyGFqXG/Q3oD4Yxru9+msvTQSd4
yWQUl+2C/wYcomhHp3pRRbbLPn4/UYxOQtOeoOP5inr9DIPzQ4Ejz744DSSR5SAN
AuYhFqN6XxOqQ04RxdlvxMQG3KIVsFGZ5IUBaY3ZplO/ul65nYgyE9jujaBBF1O6
OvEIUIHE0CwHReCsNSDZgS89lWfMDKFBcmfCP7hGfHE2sOWcTGVC38aaPjFQe76c
WIXp191xySc71qVUeuKMWBLWLwSX+jZCFxWDAQDzDYJSwjdDvtGvF1X2ddcsylki
/pK7uz5E4AuVzxoaSQf/WfSpc6VfWPIXkbPvstGq8sxx4cClylur11bpQ8ZsqzSL
EBOK2v+f2wnRGJAXy+Jq3Ur+mQhfZJiq+sM4gCCcVl9/uKnQrGWYKnZ8E9v3q0ot
O9i0/23HX9oFK8A11q6eJpXF89Y1cUwHVmAGaDoqMEc00vHYSZc+TMfzDR1PW9AD
O8wzkL1FYzSX8/5iV73PPpy1K4QCecKt6ejXg85WpEbl4HCBXotcQLL9J0+NFbaX
…
…
…
SpP2Mu79K6Rf2zZFUV4frPhuKxTtftXazLjOYbzEbIWn5l2z2knjYX8rFRZyAAD+
MVRhQqRvqK2lMH+JSYoirlA5rmiFOWeiCFayTsBE0WwA/R3ZgoH2n/lxtxyBDUmb
18Bd8Kck8P5qEEeYI+M0NbpH
=iyJr
-----END PGP PUBLIC KEY BLOCK-----
```

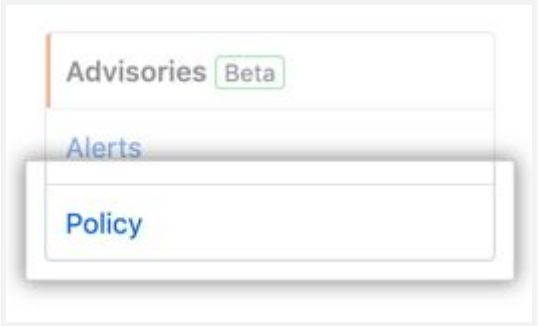
Appendix G: Setting up Github Security Policy

To set up Github Security Policy, you should:

- On GitHub, navigate to the main page of the repository.
- Under your repository name, click **Security**:



- In the left sidebar, click **Policy**:



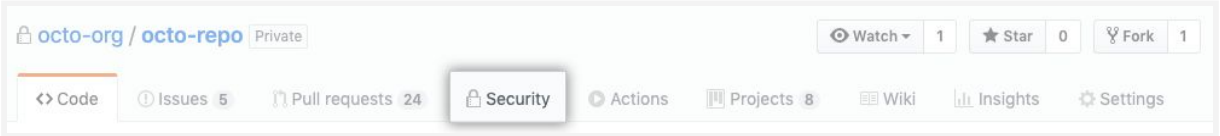
- Click **Start Setup**:

- Add and commit your SECURITY.md.

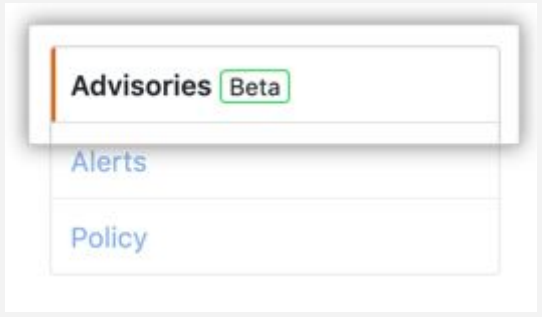
Appendix H: Setting up Maintainer Security Advisory

Anyone with admin permissions to a repository can create a security advisory:

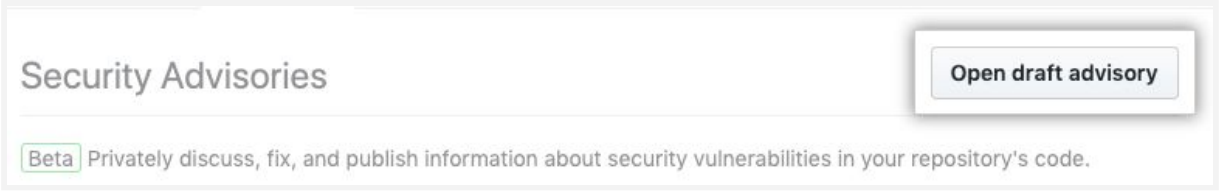
- On GitHub, navigate to the main page of the repository.
- Under your repository name, click **Security**:



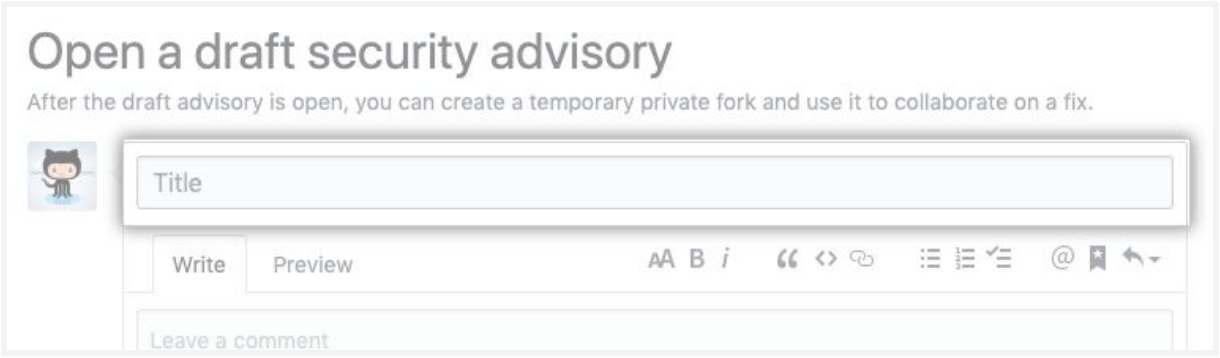
- In the left sidebar, click **Advisories:**



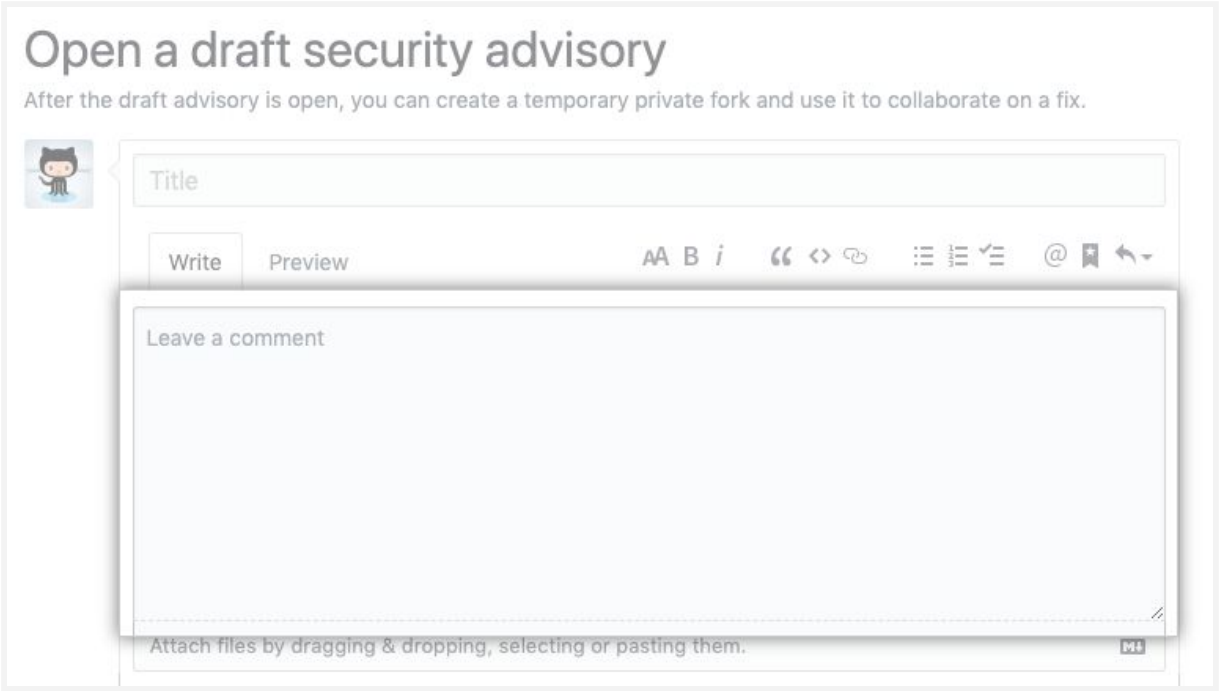
- Click **Open Draft Advisory**:



- Type a title for your security advisory:



- Type a description of the security vulnerability.



- Click **Create Draft Advisory**.