Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Yield Mountain Protocol
Platform: Polygon Network
Language: Solidity
Date: August 25th, 2022

Table of contents

Email: audit@EtherAuthority.io

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by Yield Mountain to perform the Security audit of the Yield Mountain Protocol smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on August 25th, 2022.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

Yield Mountain Protocol is a smart contract which has functions like mint, redeem, borrow, mintFresh, mintInternal, redeemVerify, borrowVerify, exitMarket, claimLoan, mintVerify, mintAllowed, seizeVerify, mintAllowed, mintVerify, seizeAllowed, seizeVerify, fail, failOpaque, delegateTo, seize, etc.

Audit scope

Name	**Code Review and Security Analysis Report for Yield Mountain Protocol Smart Contracts**
Platform	**Polygon / Solidity**
File 1	CarefulMath.sol
File 1 MD5 Hash	B0C92331384568AF33E79686796DD6DC
File 2	Comptroller.sol
File 2 MD5 Hash	6B4443E7BFDF5E7BCC4B325B5D9A2A3D
File 3	ComptrollerG1.sol
File 3 MD5 Hash	65D7D2D9C8DCF3C420F489122AF4354E
File 4	ComptrollerG2.sol
File 4 MD5 Hash	83C037E696A8B3692EFEE8EA6628DB39

File 5	ComptrollerG3.sol
File 5 MD5 Hash	FA68DDD4FAAC59E010BF3D99E5A711D0
File 6	ComptrollerG4.sol
File 6 MD5 Hash	05B23E054B52C25B9946798A0C86F6A9
File 7	ComptrollerG5.sol
File 7 MD5 Hash	CB2F0B1A4549BCB1E94C5B01185BDCEF
File 8	ComptrollerInterface.sol
File 8 MD5 Hash	3F93FDED3157FEA16B68A16DA12FAF6A
File 9	ComptrollerStorage.sol
File 9 MD5 Hash	916951C0A725180C6838E4DA6D1BA457
File 10	DAIInterestRateModelV3.sol
File 10 MD5 Hash	05CCDD2F14B39668651093D258D6426D
File 11	ErrorReporter.sol
File 11 MD5 Hash	9EC1CDFBD8FE8B80D9A19D21B4C44ADF
File 12	Exponential.sol
File 12 MD5 Hash	2374D4172B3221D95B66A0B00E6FEBB3
File 13	InterestRateModel.sol
File 13 MD5 Hash	0C89601C28A704E3689A333EAD9B5817
File 14	JumpRateModel.sol
File 14 MD5 Hash	5713635EE969D40CDC8B58A7075A0F2A
File 15	JumpRateModelV2.sol
File 15 MD5 Hash	BC8892EE020455444005B733F7C67CB7
File 16	LoanAggregatorPriceOracle.sol
File 16 MD5 Hash	A0FE5F0032AA6167ADD161E9A12E07FE
File 17	LoanPriceOracle.sol
File 17 MD5 Hash	BE7998F0F53A739C746B39040A9E72AE
File 18	Maximillion.sol

File 18 MD5 Hash	3AD8D166D3F42520FDC0CE6F2F1057E9
File 19	PriceOracle.sol
File 19 MD5 Hash	961D1806EFC7DCADD196E59343022079
File 20	Reservoir.sol
File 20 MD5 Hash	E8D1EF4974B71D15FE0C7663F66B0449
File 21	SimplePriceOracle.sol
File 21 MD5 Hash	C3D9F305392250ACEEBDE52B0DCB3D82
File 22	Timelock.sol
File 22 MD5 Hash	32F77C167FAE6E68CE3869F94A20775F
File 23	Unitroller.sol
File 23 MD5 Hash	C83330CF4111369CBEADD25BBA50557F
File 24	WhitePaperInterestRateModel.sol
File 24 MD5 Hash	3432A2971BD854F754510E5D29BDC409
File 25	YDaiDelegate.sol
File 25 MD5 Hash	691D404E0482F5DB2AED9E36DD54FEF9
File 26	YErc20.sol
File 26 MD5 Hash	00F5C920E054FD9A9275341648E3B17D
File 27	YErc20Delegate.sol
File 27 MD5 Hash	BE92BE58F502EB8D9CF29880A4A48116
File 28	YErc20Delegator.sol
File 28 MD5 Hash	F0FD98CADE00AF3D98A72369B86CB15A
File 29	YErc20Immutable.sol
File 29 MD5 Hash	6967FA2F198D77F78AB8FFDCAE9BA90E
File 30	YEther.sol
File 30 MD5 Hash	5B0096EEB9BCAAD5631F55233CAE5006
File 31	YLoanLikeDelegate.sol
File 31 MD5 Hash	CE78623AC23E00AC2A5050820709D690

File 32	YToken.sol
File 32 MD5 Hash	C3CE5CF725FD2180DE5D9C6834035601
File 33	YTokenInterfaces.sol
File 33 MD5 Hash	66FB5E914992FB76A383765AFB32BBE9
File 34	GovernorAlpha.sol
File 34 MD5 Hash	4FE3C666BE924E347032EA59E192CCB3
File 35	LOAN.sol
File 35 MD5 Hash	6A2BCE0F39C37CCBD4E21BA8D759B347
File 36	LoanLens.sol
File 36 MD5 Hash	631B3C96840AA0CB2EA2B2FAEA3275F9
File 37	LoanReserve.sol
File 37 MD5 Hash	3920F7336E39019F9633B4F707433C5D
File 38	LoanStaking.sol
File 38 MD5 Hash	361E9450FF39B3573CB35535A5F238C3
File 39	LoanStakingProxy.sol
File 39 MD5 Hash	DC659EB4D101C602995DBFD2447C1A73
File 40	NFTController.sol
File 40 MD5 Hash	6A311B61EF9805A580C604B354989100
Audit Date	August 25th,2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 CarefulMath.sol • CarefulMath has functions like: mulUInt, divUInt, subUInt, etc.	**YES, This is valid.**
File 2 Comptroller.sol • Loan Claim Threshold: 0.001 • Close Factor Minimum Mantissa: 0.05 • Close Factor Mantissa: 0.9 • Collateral Factor Maximum Mantissa: 0.9 • Liquidation Incentive Mantissa: 1.0 • Liquidation Incentive Maximum Mantissa: 1.5	**YES, This is valid.**
File 3 ComptrollerG1.sol • Close Factor Minimum Mantissa: 0.05 • Close Factor Maximum Mantissa: 0.9 • Collateral Factor Maximum Mantissa: 0.9 • Liquidation Incentive Maximum Mantissa: 1.5	**YES, This is valid.**
File 4 ComptrollerG2.sol • Close Factor Minimum Mantissa: 0.05 • Close Factor Maximum Mantissa: 0.9 • Collateral Factor Maximum Mantissa: 0.9 • Liquidation Incentive Minimum Mantissa: 1.0 • Liquidation Incentive Maximum Mantissa: 1.5	**YES, This is valid.**
File 5 ComptrollerG3.sol • Close Factor Minimum Mantissa: 0.05 • Close Factor Maximum Mantissa: 0.9 • Collateral Factor Maximum Mantissa: 0.9 • Liquidation Incentive Minimum Mantissa: 1.0 • Liquidation Incentive Maximum Mantissa: 1.5	**YES, This is valid.**

File 6 ComptrollerG4.sol • Close Factor Minimum Mantissa: 0.05 • Close Factor Maximum Mantissa: 0.9 • Collateral Factor Maximum Mantissa: 0.9 • Liquidation Incentive Minimum Mantissa: 1.0 • Liquidation Incentive Maximum Mantissa: 1.5	**YES, This is valid.**
File 7 ComptrollerG5.sol • Close Factor Minimum Mantissa: 0.05 • Close Factor Maximum Mantissa: 0.9 • Collateral Factor Maximum Mantissa: 0.9 • Liquidation Incentive Minimum Mantissa: 1.0 • Liquidation Incentive Maximum Mantissa: 1.5	**YES, This is valid.**
File 8 ComptrollerInterface.sol • ComptrollerInterface has functions like: mintAllowed, mintVerify, redeemAllowed, etc.	**YES, This is valid.**
File 9 ComptrollerStorage.sol •	**YES, This is valid.**
File 10 DAIInterestRateModelV3.sol • Assumed One Minus Reserve Factor Mantissa: 0.95 .	**YES, This is valid.**
File 11 ErrorReporter.sol • ErrorReporter has functions like: failOpaque, fail, etc.	**YES, This is valid.**
File 12 Exponential.sol • Exponential module for storing fixed-precision decimals.	**YES, This is valid.**
File 13 InterestRateModel.sol • InterestRateModel has functions like: getBorrowRate, etc.	**YES, This is valid.**

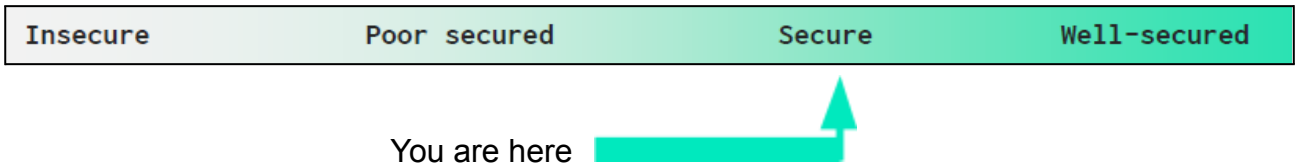
File 14 JumpRateModel.sol • JumpRateModel has functions like: utilizationRate, getBorrowRate, etc.	**YES, This is valid.**
File 15 JumpRateModelV2.sol • JumpRateModelV2 has functions like: utilizationRate, updateJumpRateModel, etc.	**YES, This is valid.**
File 16 LoanAggregatorPriceOracle.sol • LoanAggregatorPriceOracle owner can set Underlying Price, direct price.	**YES, This is valid.**
File 17 LoanPriceOracle.sol • LoanPriceOracle owner can set reference address, Underlying Price, direct price, etc.	**YES, This is valid.**
File 18 Maximillion.sol • Maximillion has functions like: repayBehalfExplicit, etc.	**YES, This is valid.**
File 19 PriceOracle.sol • PriceOracle has functions like: getUnderlyingPrice.	**YES, This is valid.**
File 20 Reservoir.sol • Reservoir has functions like: drip	**YES, This is valid.**
File 21 SimplePriceOracle.sol • SimplePriceOracle owners can set UnderlyingPrice, DirectPrice, etc.	**YES, This is valid.**
File 22 Timelock.sol • Grace Period: 14 Days • Minimum Delay: 2 Days • Maximum Delay: 30 Days	**YES, This is valid.**
File 23 Unitroller.sol • Unitroller admin can set pending implementation,	**YES, This is valid.**

accept admin, etc.	
File 24 WhitePaperInterestRateModel.sol • WhitePaperInterestRateModel has functions like: utilizationRate, getBorrowRate, etc.	**YES, This is valid.**
File 25 YDaiDelegate.sol • YDaiDelegate has functions like: _becomeImplementation, accrueInterest, etc.	**YES, This is valid.**
File 26 YErc20.sol • YErc20 has functions like: mint, redeem, etc.	**YES, This is valid.**
File 27 YErc20Delegate.sol • YErc20Delegate has functions like: _resignImplementation, etc.	**YES, This is valid.**
File 28 YErc20Delegator.sol • YErc20Delegator has functions like: _setImplementation, mint, etc.	**YES, This is valid.**
File 29 YErc20Immutable.sol •	**YES, This is valid.**
File 30 YEther.sol • YEther has functions like: mint, redeem, etc.	**YES, This is valid.**
File 31 YLoanLikeDelegate.sol • YLoanLikeDelegate has functions like: _delegateLoanLikeTo.	**YES, This is valid.**
File 32 YToken.sol • YToken has functions like: initialize, transferTokens, etc.	**YES, This is valid.**
File 33 YTokenInterfaces.sol • YTokenInterface owners can set interest rate	**YES, This is valid.**

model, reduce reserve amounts.	
File 34 GovernorAlpha.sol • Name: Loan Governor Alpha	**YES, This is valid.**
File 35 LOAN.sol • Name: Yield Mountain • Symbol: LOAN • Decimals: 18 • Total Supply: 200 Million LOAN	**YES, This is valid.**
File 36 LoanLens.sol • LoanLens has functions like: yTokenMetadata, yTokenBalances, etc.	**YES, This is valid.**
File 37 LoanReserve.sol • LoanReserve has functions like: setLoan, setRewarder, etc.	**YES, This is valid.**
File 38 LoanStaking.sol • rewardsDuration: 1 week • lockDuration: 4 weeks	**YES, This is valid.**
File 39 NFTController.sol • NFTController has functions like: setWhitelist, setDefaultBoostRate, etc.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. Also, these contracts do contain owner control, which does not make them fully decentralized.



We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 2 low and some very low level issues.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Moderated
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Moderated
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Moderated
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Email: audit@EtherAuthority.io

Code Quality

This audit scope has 40 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in the Yield Mountain Protocol are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Yield Mountain Protocol.

The Yield Mountain team has not provided unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Some code parts are not well commented on smart contracts. We suggest using Ethereum's NatSpec style for the commenting.

Documentation

We were given a Yield Mountain Protocol smart contract code in the form of a file. The hash of that code is mentioned above in the table.

As mentioned above, code parts are not well commented. But the logic is straightforward. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Use of Dependencies

As per our observation, the libraries are used in this smart contracts infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

CarefulMath.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	mulUInt	internal	Passed	No Issue
3	divUInt	internal	Passed	No Issue
4	subUInt	internal	Passed	No Issue
5	addUInt	internal	Passed	No Issue
6	addThenSubUInt	internal	Passed	No Issue

Comptroller.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getAssetsIn	external	Passed	No Issue
3	checkMembership	external	Passed	No Issue
4	enterMarkets	write	High consumption 'for/while' loop	Refer Audit Findings
5	addToMarketInternal	internal	Passed	No Issue
6	exitMarket	external	Passed	No Issue
7	mintAllowed	external	Passed	No Issue
8	mintVerify	external	Passed	No Issue
9	redeemAllowed	external	Passed	No Issue
10	redeemAllowedInternal	internal	Passed	No Issue
11	redeemVerify	external	Passed	No Issue
12	borrowAllowed	external	Passed	No Issue
13	borrowVerify	external	Passed	No Issue
14	repayBorrowAllowed	external	Passed	No Issue
15	repayBorrowVerify	external	Passed	No Issue
16	liquidateBorrowAllowed	external	Passed	No Issue
17	liquidateBorrowVerify	external	Passed	No Issue
18	seizeAllowed	external	Passed	No Issue
19	seizeVerify	external	Passed	No Issue
20	transferAllowed	external	Passed	No Issue
21	transferVerify	external	Passed	No Issue
22	getAccountLiquidity	read	Passed	No Issue
23	getAccountLiquidityInternal	internal	Passed	No Issue
24	getHypotheticalAccountLiquidity	read	Passed	No Issue

25	getHypotheticalAccountLiquidityInternal	internal	Passed	No Issue
26	liquidateCalculateSeizeTokens	external	Passed	No Issue
27	_setPriceOracle	write	Passed	No Issue
28	_setCloseFactor	external	Passed	No Issue
29	_setCollateralFactor	external	Passed	No Issue
30	_setMaxAssets	external	Passed	No Issue
31	_setLiquidationIncentive	external	Passed	No Issue
32	_supportMarket	external	Passed	No Issue
33	_initializeMarket	internal	Passed	No Issue
34	_addMarketInternal	internal	Passed	No Issue
35	_setPauseGuardian	write	Passed	No Issue
36	_setMintPaused	write	Passed	No Issue
37	_setBorrowPaused	write	Passed	No Issue
38	_setTransferPaused	write	Passed	No Issue
39	_setSeizePaused	write	Passed	No Issue
40	_setReserveInfo	external	Passed	No Issue
41	_become	write	Passed	No Issue
42	adminOrInitializing	internal	Passed	No Issue
43	_setLoanSpeeds	write	Passed	No Issue
44	setLoanSpeedInternal	internal	Passed	No Issue
45	updateLoanSupplyIndex	internal	Passed	No Issue
46	updateLoanBorrowIndex	internal	Passed	No Issue
47	distributeSupplierLoan	internal	Passed	No Issue
48	distributeBorrowerLoan	internal	Passed	No Issue
49	transferLoan	internal	Passed	No Issue
50	claimLoan	write	Passed	No Issue
51	claimLoan	write	Passed	No Issue
52	claimLoan	write	Passed	No Issue
53	grantLOANInternal	internal	Passed	No Issue
54	updateContributorRewards	write	Passed	No Issue
55	_setContributorLoanSpeed	write	Passed	No Issue
56	_grantLOAN	write	Passed	No Issue
57	_setLoanRate	write	Loan Rate has no limit	Refer Audit Findings
58	_dropLoanMarket	write	Passed	No Issue
59	getAllMarkets	read	Passed	No Issue
60	getBlockNumber	read	Passed	No Issue
61	getLOANAddress	read	Passed	No Issue
62	canClaimLoanBySuppling	read	Passed	No Issue

ComptrollerG1.sol

Functions

Sl.	Functions	Type	Observation	Conclusion

1	constructor	write	Passed	No Issue
2	getAssetsIn	external	Passed	No Issue
3	checkMembership	external	Passed	No Issue
4	enterMarkets	write	Passed	No Issue
5	exitMarket	external	Passed	No Issue
6	mintAllowed	external	Passed	No Issue
7	mintVerify	external	Passed	No Issue
8	redeemAllowed	external	Passed	No Issue
9	redeemAllowedInternal	internal	Passed	No Issue
10	redeemVerify	external	Passed	No Issue
11	borrowAllowed	external	Passed	No Issue
12	borrowVerify	external	Passed	No Issue
13	repayBorrowAllowed	external	Passed	No Issue
14	repayBorrowVerify	external	Passed	No Issue
15	liquidateBorrowAllowed	external	Passed	No Issue
16	liquidateBorrowVerify	external	Passed	No Issue
17	seizeAllowed	external	Passed	No Issue
18	seizeVerify	external	Passed	No Issue
19	transferAllowed	external	Passed	No Issue
20	transferVerify	external	Passed	No Issue
21	getAccountLiquidity	read	Passed	No Issue
22	getAccountLiquidityInternal	internal	Passed	No Issue
23	getHypotheticalAccountLiquidity	read	Passed	No Issue
24	getHypotheticalAccountLiquidityInternal	internal	Passed	No Issue
25	liquidateCalculateSeizeTokens	external	Passed	No Issue
26	_setPriceOracle	write	Passed	No Issue
27	_setCloseFactor	external	Passed	No Issue
28	_setCollateralFactor	external	Passed	No Issue
29	_setMaxAssets	external	Passed	No Issue
30	_setLiquidationIncentive	external	Passed	No Issue
31	_supportMarket	external	Passed	No Issue
32	_become	write	Passed	No Issue
33	adminOrInitializing	internal	Passed	No Issue

ComptrollerG2.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getAssetsIn	external	Passed	No Issue
3	checkMembership	external	Passed	No Issue
4	enterMarkets	write	Passed	No Issue

5	exitMarket	external	Passed	No Issue
6	mintAllowed	external	Passed	No Issue
7	mintVerify	external	Passed	No Issue
8	redeemAllowed	external	Passed	No Issue
9	redeemAllowedInternal	internal	Passed	No Issue
10	redeemVerify	external	Passed	No Issue
11	borrowAllowed	external	Passed	No Issue
12	borrowVerify	external	Passed	No Issue
13	repayBorrowAllowed	external	Passed	No Issue
14	repayBorrowVerify	external	Passed	No Issue
15	liquidateBorrowAllowed	external	Passed	No Issue
16	liquidateBorrowVerify	external	Passed	No Issue
17	seizeAllowed	external	Passed	No Issue
18	seizeVerify	external	Passed	No Issue
19	transferAllowed	external	Passed	No Issue
20	transferVerify	external	Passed	No Issue
21	getAccountLiquidity	read	Passed	No Issue
22	getAccountLiquidityInternal	internal	Passed	No Issue
23	getHypotheticalAccountLiquidity	read	Passed	No Issue
24	getHypotheticalAccountLiquidityInternal	internal	Passed	No Issue
25	liquidateCalculateSeizeTokens	external	Passed	No Issue
26	_setPriceOracle	write	Passed	No Issue
27	_setCloseFactor	external	Passed	No Issue
28	_setCollateralFactor	external	Passed	No Issue
29	_setMaxAssets	external	Passed	No Issue
30	_setLiquidationIncentive	external	Passed	No Issue
31	_supportMarket	external	Passed	No Issue
32	_setPauseGuardian	write	Passed	No Issue
33	_setMintPaused	write	Passed	No Issue
34	addToMarketInternal	internal	Passed	No Issue
35	getHypotheticalAccountLiquidity	read	Passed	No Issue
36	_setBorrowPaused	write	Passed	No Issue
37	_setTransferPaused	write	Passed	No Issue
38	_setSeizePaused	write	Passed	No Issue
39	_become	write	Passed	No Issue

ComptrollerG3.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getAssetsIn	external	Passed	No Issue

3	checkMembership	external	Passed	No Issue
4	enterMarkets	write	Passed	No Issue
5	exitMarket	external	Passed	No Issue
6	mintAllowed	external	Passed	No Issue
7	mintVerify	external	Passed	No Issue
8	redeemAllowed	external	Passed	No Issue
9	redeemAllowedInternal	internal	Passed	No Issue
10	redeemVerify	external	Passed	No Issue
11	borrowAllowed	external	Passed	No Issue
12	borrowVerify	external	Passed	No Issue
13	repayBorrowAllowed	external	Passed	No Issue
14	repayBorrowVerify	external	Passed	No Issue
15	liquidateBorrowAllowed	external	Passed	No Issue
16	liquidateBorrowVerify	external	Passed	No Issue
17	seizeAllowed	external	Passed	No Issue
18	seizeVerify	external	Passed	No Issue
19	transferAllowed	external	Passed	No Issue
20	transferVerify	external	Passed	No Issue
21	getAccountLiquidity	read	Passed	No Issue
22	getAccountLiquidityInternal	internal	Passed	No Issue
23	getHypotheticalAccountLiquidity	read	Passed	No Issue
24	getHypotheticalAccountLiquidityInternal	internal	Passed	No Issue
25	liquidateCalculateSeizeTokens	external	Passed	No Issue
26	_setPriceOracle	write	Passed	No Issue
27	_setCloseFactor	external	Passed	No Issue
28	_setCollateralFactor	external	Passed	No Issue
29	_setMaxAssets	external	Passed	No Issue
30	_setLiquidationIncentive	external	Passed	No Issue
31	_supportMarket	external	Passed	No Issue
32	_setPauseGuardian	write	Passed	No Issue
33	_setMintPaused	write	Passed	No Issue
34	addToMarketInternal	internal	Passed	No Issue
35	getHypotheticalAccountLiquidity	read	Passed	No Issue
36	_setBorrowPaused	write	Passed	No Issue
37	_setTransferPaused	write	Passed	No Issue
38	_setSeizePaused	write	Passed	No Issue
39	_become	write	Passed	No Issue
40	_addMarketInternal	internal	Passed	No Issue
41	_becomeG3	write	Passed	No Issue
42	adminOrInitializing	internal	Passed	No Issue
43	refreshLoanSpeeds	write	Passed	No Issue
44	updateLoanSupplyIndex	internal	Passed	No Issue
45	updateLoanBorrowIndex	internal	Passed	No Issue

46	distributeSupplierLoan	internal	Passed	No Issue
47	distributeBorrowerLoan	internal	Passed	No Issue
48	transferLoan	internal	Passed	No Issue
49	claimLoan	write	Passed	No Issue
50	claimLoan	write	Passed	No Issue
51	claimLoan	write	Passed	No Issue
52	_setLoanRate	write	Passed	No Issue
53	_addLoanMarkets	write	Passed	No Issue
54	_addLoanMarketInternal	internal	Passed	No Issue
55	_dropLoanMarket	write	Passed	No Issue
56	getAllMarkets	read	Passed	No Issue
57	getBlockNumber	read	Passed	No Issue
58	getLOANAddress	read	Passed	No Issue

ComptrollerG4.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getAssetsIn	external	Passed	No Issue
3	checkMembership	external	Passed	No Issue
4	enterMarkets	write	Passed	No Issue
5	exitMarket	external	Passed	No Issue
6	mintAllowed	external	Passed	No Issue
7	mintVerify	external	Passed	No Issue
8	redeemAllowed	external	Passed	No Issue
9	redeemAllowedInternal	internal	Passed	No Issue
10	redeemVerify	external	Passed	No Issue
11	borrowAllowed	external	Passed	No Issue
12	borrowVerify	external	Passed	No Issue
13	repayBorrowAllowed	external	Passed	No Issue
14	repayBorrowVerify	external	Passed	No Issue
15	liquidateBorrowAllowed	external	Passed	No Issue
16	liquidateBorrowVerify	external	Passed	No Issue
17	seizeAllowed	external	Passed	No Issue
18	seizeVerify	external	Passed	No Issue
19	transferAllowed	external	Passed	No Issue
20	transferVerify	external	Passed	No Issue
21	getAccountLiquidity	read	Passed	No Issue
22	getAccountLiquidityInternal	internal	Passed	No Issue
23	getHypotheticalAccountLiquidity	read	Passed	No Issue
24	getHypotheticalAccountLiquidityInternal	internal	Passed	No Issue
25	liquidateCalculateSeizeTokens	external	Passed	No Issue
26	_setPriceOracle	write	Passed	No Issue

27	_setCloseFactor	external	Passed	No Issue
28	_setCollateralFactor	external	Passed	No Issue
29	_setMaxAssets	external	Passed	No Issue
30	_setLiquidationIncentive	external	Passed	No Issue
31	_supportMarket	external	Passed	No Issue
32	_setPauseGuardian	write	Passed	No Issue
33	_setMintPaused	write	Passed	No Issue
34	addToMarketInternal	internal	Passed	No Issue
35	getHypotheticalAccountLiquidity	read	Passed	No Issue
36	_setBorrowPaused	write	Passed	No Issue
37	_setTransferPaused	write	Passed	No Issue
38	_setSeizePaused	write	Passed	No Issue
39	_become	write	Passed	No Issue
40	_addMarketInternal	internal	Passed	No Issue
41	grantLOANInternal	internal	Passed	No Issue
42	adminOrInitializing	internal	Passed	No Issue
43	updateLoanSupplyIndex	internal	Passed	No Issue
44	updateLoanBorrowIndex	internal	Passed	No Issue
45	distributeSupplierLoan	internal	Passed	No Issue
46	distributeBorrowerLoan	internal	Passed	No Issue
47	transferLoan	internal	Passed	No Issue
48	claimLoan	write	Passed	No Issue
49	claimLoan	write	Passed	No Issue
50	claimLoan	write	Passed	No Issue
51	_setLoanRate	write	Passed	No Issue
52	_addLoanMarkets	write	Passed	No Issue
53	_addLoanMarketInternal	internal	Passed	No Issue
54	_dropLoanMarket	write	Passed	No Issue
55	getAllMarkets	read	Passed	No Issue
56	getBlockNumber	read	Passed	No Issue
57	getLOANAddress	read	Passed	No Issue

ComptrollerG5.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getAssetsIn	external	Passed	No Issue
3	checkMembership	external	Passed	No Issue
4	enterMarkets	write	Passed	No Issue
5	exitMarket	external	Passed	No Issue

6	mintAllowed	external	Passed	No Issue
7	mintVerify	external	Passed	No Issue
8	redeemAllowed	external	Passed	No Issue
9	redeemAllowedInternal	internal	Passed	No Issue
10	redeemVerify	external	Passed	No Issue
11	borrowAllowed	external	Passed	No Issue
12	borrowVerify	external	Passed	No Issue
13	repayBorrowAllowed	external	Passed	No Issue
14	repayBorrowVerify	external	Passed	No Issue
15	liquidateBorrowAllowed	external	Passed	No Issue
16	liquidateBorrowVerify	external	Passed	No Issue
17	seizeAllowed	external	Passed	No Issue
18	seizeVerify	external	Passed	No Issue
19	transferAllowed	external	Passed	No Issue
20	transferVerify	external	Passed	No Issue
21	getAccountLiquidity	read	Passed	No Issue
22	getAccountLiquidityInternal	internal	Passed	No Issue
23	getHypotheticalAccountLiquidity	read	Passed	No Issue
24	getHypotheticalAccountLiquidityInternal	internal	Passed	No Issue
25	liquidateCalculateSeizeTokens	external	Passed	No Issue
26	_setPriceOracle	write	Passed	No Issue
27	_setCloseFactor	external	Passed	No Issue
28	_setCollateralFactor	external	Passed	No Issue
29	_setMaxAssets	external	Passed	No Issue
30	_setLiquidationIncentive	external	Passed	No Issue
31	_supportMarket	external	Passed	No Issue
32	_setPauseGuardian	write	Passed	No Issue
33	_setMintPaused	write	Passed	No Issue
34	addToMarketInternal	internal	Passed	No Issue
35	getHypotheticalAccountLiquidity	read	Passed	No Issue
36	_setBorrowPaused	write	Passed	No Issue
37	_setTransferPaused	write	Passed	No Issue
38	_setSeizePaused	write	Passed	No Issue
39	_become	write	Passed	No Issue
40	_addMarketInternal	internal	Passed	No Issue
41	grantLOANInternal	internal	Passed	No Issue
42	adminOrInitializing	internal	Passed	No Issue
43	updateLoanSupplyIndex	internal	Passed	No Issue
44	updateLoanBorrowIndex	internal	Passed	No Issue
45	distributeSupplierLoan	internal	Passed	No Issue
46	distributeBorrowerLoan	internal	Passed	No Issue
47	transferLoan	internal	Passed	No Issue
48	claimLoan	write	Passed	No Issue
49	claimLoan	write	Passed	No Issue

50	claimLoan	write	Passed	No Issue
51	_setLoanRate	write	Passed	No Issue
52	updateContributorRewards	write	Passed	No Issue
53	_initializeMarket	internal	Passed	No Issue
54	_dropLoanMarket	write	Passed	No Issue
55	getAllMarkets	read	Passed	No Issue
56	getBlockNumber	read	Passed	No Issue
57	getLOANAddress	read	Passed	No Issue
58	canClaimLoanBySuppling	read	Passed	No Issue
59	_grantLOAN	write	Passed	No Issue
60	_setContributorLoanSpeed	write	Passed	No Issue

ComptrollerInterface.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	enterMarkets	external	Passed	No Issue
3	exitMarket	external	Passed	No Issue
4	mintAllowed	external	Passed	No Issue
5	mintVerify	external	Passed	No Issue
6	redeemAllowed	external	Passed	No Issue
7	redeemVerify	external	Passed	No Issue
8	borrowAllowed	external	Passed	No Issue
9	borrowVerify	external	Passed	No Issue
10	repayBorrowAllowed	external	Passed	No Issue
11	repayBorrowVerify	external	Passed	No Issue
12	liquidateBorrowAllowed	external	Passed	No Issue
13	liquidateBorrowVerify	external	Passed	No Issue
14	seizeAllowed	external	Passed	No Issue
15	seizeVerify	external	Passed	No Issue
16	transferAllowed	external	Passed	No Issue
17	transferVerify	external	Passed	No Issue
18	liquidateCalculateSeizeTokens	external	Passed	No Issue

DAIInterestRateModelV3.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	updateJumpRateModel	write	Passed	No Issue
3	getSupplyRate	read	Passed	No Issue
4	dsrPerBlock	read	Passed	No Issue

5	poke	write	Division before multiplication	Refer Audit Findings

ErrorReporter.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	fail	internal	Passed	No Issue
3	failOpaque	internal	Passed	No Issue

Exponential.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getExp	internal	Passed	No Issue
3	addExp	internal	Passed	No Issue
4	subExp	internal	Passed	No Issue
5	mulScalar	internal	Passed	No Issue
6	mulScalarTruncate	internal	Passed	No Issue
7	mulScalarTruncateAddUInt	internal	Passed	No Issue
8	divScalar	internal	Passed	No Issue
9	divScalarByExp	internal	Passed	No Issue
10	divScalarByExpTruncate	internal	Passed	No Issue
11	mulExp	internal	Passed	No Issue
12	mulExp	internal	Passed	No Issue
13	mulExp3	internal	Passed	No Issue
14	divExp	internal	Passed	No Issue
15	truncate	internal	Passed	No Issue
16	lessThanExp	internal	Passed	No Issue
17	lessThanOrEqualExp	internal	Passed	No Issue
18	greaterThanExp	internal	Passed	No Issue
19	isZeroExp	internal	Passed	No Issue
20	safe224	internal	Passed	No Issue
21	safe32	internal	Passed	No Issue
22	add	internal	Passed	No Issue
23	add_	write	Passed	No Issue
24	sub_	write	Passed	No Issue
25	mul_	write	Passed	No Issue
26	div_	write	Passed	No Issue
27	fraction	internal	Passed	No Issue

InterestRateModel.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getBorrowRate	external	Passed	No Issue
3	getSupplyRate	external	Passed	No Issue

JumpRateModel.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	utilizationRate	write	Passed	No Issue
3	getBorrowRate	read	Passed	No Issue
4	getSupplyRate	read	Passed	No Issue

JumpRateModelV2.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	updateJumpRateModel	external	Passed	No Issue
3	utilizationRate	write	Passed	No Issue
4	getBorrowRate	read	Passed	No Issue
5	getSupplyRate	read	Passed	No Issue
6	updateJumpRateModelInternal	internal	Passed	No Issue

LoanAggregatorPriceOracle.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getUnderlyingPrice	read	Passed	No Issue
3	setUnderlyingPrice	write	Passed	No Issue
4	setDirectPrice	write	Passed	No Issue
5	getChainlinkPrice	internal	Passed	No Issue
6	setFeed	external	Passed	No Issue
7	setRef	external	Passed	No Issue
8	getFeed	read	Passed	No Issue
9	assetPrices	external	Passed	No Issue

10	compareStrings	internal	Passed	No Issue
11	setAdmin	external	Passed	No Issue

LoanPriceOracle.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	setRefAddress	write	Passed	No Issue
3	getRefAddress	read	Passed	No Issue
4	getUnderlyingPrice	read	Passed	No Issue
5	setUnderlyingPrice	write	Passed	No Issue
6	setDirectPrice	write	Passed	No Issue
7	assetPrices	external	Passed	No Issue
8	compareStrings	internal	Passed	No Issue
9	setAdmin	external	Passed	No Issue

Maximillion.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	repayBehalf	write	Passed	No Issue
3	repayBehalfExplicit	write	Passed	No Issue

PriceOracle.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getUnderlyingPrice	external	Passed	No Issue

Reservoir.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	drip	write	Passed	No Issue
3	add	internal	Passed	No Issue
4	sub	internal	Passed	No Issue
5	mul	internal	Passed	No Issue
6	min	internal	Passed	No Issue

SimplePriceOracle.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getUnderlyingPrice	read	Passed	No Issue
3	setUnderlyingPrice	write	Passed	No Issue
4	setDirectPrice	write	Passed	No Issue
5	assetPrices	external	Passed	No Issue
6	compareStrings	internal	Passed	No Issue

Timelock.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	setDelay	write	Passed	No Issue
3	acceptAdmin	write	Passed	No Issue
4	setPendingAdmin	write	Passed	No Issue
5	queueTransaction	write	Passed	No Issue
6	cancelTransaction	write	Passed	No Issue
7	executeTransaction	write	Passed	No Issue
8	getBlockTimestamp	internal	Passed	No Issue

Unitroller.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_acceptImplementation	write	Passed	No Issue
3	_setPendingImplementation	write	Passed	No Issue
4	_setPendingAdmin	write	Passed	No Issue
5	_acceptAdmin	write	Passed	No Issue

WhitePaperInterestRateModel.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue

2	utilizationRate	write	Passed	No Issue
3	getBorrowRate	read	Passed	No Issue
4	getSupplyRate	read	Passed	No Issue

YDaiDelegate.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_becomeImplementation	write	Passed	No Issue
3	_becomeImplementation	internal	Passed	No Issue
4	_resignImplementation	write	Passed	No Issue
5	accrueInterest	write	Passed	No Issue
6	getCashPrior	internal	Passed	No Issue
7	doTransferIn	internal	Passed	No Issue
8	doTransferOut	internal	Passed	No Issue
9	add	internal	Passed	No Issue
10	mul	internal	Passed	No Issue

YErc20.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initialize	write	Passed	No Issue
3	mint	external	Passed	No Issue
4	redeem	external	Passed	No Issue
5	redeemUnderlying	external	Passed	No Issue
6	borrow	external	Passed	No Issue
7	repayBorrow	external	Passed	No Issue
8	repayBorrowBehalf	external	Passed	No Issue
9	liquidateBorrow	external	Passed	No Issue
10	_addReserves	external	Passed	No Issue
11	getCashPrior	internal	Passed	No Issue
12	doTransferIn	internal	Passed	No Issue
13	doTransferOut	internal	Passed	No Issue

YErc20Delegate.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_becomeImplementation	write	Passed	No Issue

3	_resignImplementation	write	Passed	No Issue

YErc20Delegator.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_setImplementation	write	Passed	No Issue
3	mint	external	Passed	No Issue
4	redeem	external	Passed	No Issue
5	redeemUnderlying	external	Passed	No Issue
6	borrow	external	Passed	No Issue
7	repayBorrow	external	Passed	No Issue
8	repayBorrowBehalf	external	Passed	No Issue
9	liquidateBorrow	external	Passed	No Issue
10	transfer	external	Passed	No Issue
11	transferFrom	external	Passed	No Issue
12	approve	external	Passed	No Issue
13	allowance	external	Passed	No Issue
14	balanceOf	external	Passed	No Issue
15	balanceOfUnderlying	external	Passed	No Issue
16	getAccountSnapshot	external	Passed	No Issue
17	borrowRatePerBlock	external	Passed	No Issue
18	supplyRatePerBlock	external	Passed	No Issue
19	totalBorrowsCurrent	external	Passed	No Issue
20	borrowBalanceCurrent	external	Passed	No Issue
21	borrowBalanceStored	read	Passed	No Issue
22	exchangeRateCurrent	write	Passed	No Issue
23	exchangeRateStored	read	Passed	No Issue
24	getCash	external	Passed	No Issue
25	accrueInterest	write	Passed	No Issue
26	seize	external	Passed	No Issue
27	_setPendingAdmin	external	Passed	No Issue
28	_setComptroller	write	Passed	No Issue
29	_setReserveFactor	external	Passed	No Issue
30	_acceptAdmin	external	Passed	No Issue
31	_addReserves	external	Passed	No Issue
32	_reduceReserves	external	Passed	No Issue
33	_transferReserves	external	Passed	No Issue
34	_setInterestRateModel	write	Passed	No Issue
35	delegateTo	internal	Passed	No Issue
36	delegateToImplementation	write	Passed	No Issue
37	delegateToViewImplementation	read	Passed	No Issue

YErc20Immutable.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue

YEther.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	mint	external	Passed	No Issue
3	redeem	external	Passed	No Issue
4	redeemUnderlying	external	Passed	No Issue
5	borrow	external	Passed	No Issue
6	repayBorrow	external	Passed	No Issue
7	repayBorrowBehalf	external	Passed	No Issue
8	liquidateBorrow	external	Passed	No Issue
9	getCashPrior	internal	Passed	No Issue
10	doTransferIn	internal	Passed	No Issue
11	doTransferOut	internal	Passed	No Issue
12	requireNoError	internal	Passed	No Issue

YLoanLikeDelegate.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_delegateLoanLikeTo	external	Passed	No Issue

YToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initialize	write	Passed	No Issue
3	transferTokens	internal	Passed	No Issue
4	transfer	external	Passed	No Issue
5	transferFrom	external	Passed	No Issue
6	approve	external	Passed	No Issue
7	allowance	external	Passed	No Issue
8	balanceOf	external	Passed	No Issue

9	balanceOfUnderlying	external	Passed	No Issue
10	getAccountSnapshot	external	Passed	No Issue
11	getBlockNumber	internal	Passed	No Issue
12	borrowRatePerBlock	external	Passed	No Issue
13	supplyRatePerBlock	external	Passed	No Issue
14	totalBorrowsCurrent	external	Passed	No Issue
15	borrowBalanceCurrent	external	Passed	No Issue
16	borrowBalanceStored	read	Passed	No Issue
17	borrowBalanceStoredInternal	internal	Passed	No Issue
18	exchangeRateCurrent	write	Passed	No Issue
19	exchangeRateStored	read	Passed	No Issue
20	exchangeRateStoredInternal	internal	Passed	No Issue
21	getCash	external	Passed	No Issue
22	accrueInterest	write	Passed	No Issue
23	mintInternal	internal	Passed	No Issue
24	mintFresh	internal	Passed	No Issue
25	redeemInternal	internal	Passed	No Issue
26	redeemUnderlyingInternal	internal	Passed	No Issue
27	redeemFresh	internal	Passed	No Issue
28	borrowInternal	internal	Passed	No Issue
29	borrowFresh	internal	Passed	No Issue
30	repayBorrowInternal	internal	Passed	No Issue
31	repayBorrowBehalfInternal	internal	Passed	No Issue
32	repayBorrowFresh	internal	Passed	No Issue
33	liquidateBorrowInternal	internal	Passed	No Issue
34	liquidateBorrowFresh	internal	Passed	No Issue
35	seize	external	Passed	No Issue
36	seizeInternal	internal	Passed	No Issue
37	_setPendingAdmin	external	Passed	No Issue
38	_acceptAdmin	external	Passed	No Issue
39	_setComptroller	write	Passed	No Issue
40	_setReserveFactor	external	Passed	No Issue
41	_setReserveFactorFresh	internal	Passed	No Issue
42	_addReservesInternal	internal	Passed	No Issue
43	_addReservesFresh	internal	Passed	No Issue
44	_reduceReserves	external	Passed	No Issue
45	_reduceReservesFresh	internal	Passed	No Issue
46	_transferReserves	external	Passed	No Issue
47	_transferReservesFresh	internal	Passed	No Issue
48	_setInterestRateModel	write	Passed	No Issue
49	_setInterestRateModelFresh	internal	Passed	No Issue
50	getCashPrior	internal	Passed	No Issue
51	doTransferIn	internal	Passed	No Issue
52	doTransferOut	internal	Passed	No Issue
53	nonReentrant	modifier	Passed	No Issue

YTokenInterface.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	transfer	external	Passed	No Issue
3	transferFrom	external	Passed	No Issue
4	approve	external	Passed	No Issue
5	allowance	external	Passed	No Issue
6	balanceOf	external	Passed	No Issue
7	balanceOfUnderlying	external	Passed	No Issue
8	getAccountSnapshot	external	Passed	No Issue
9	borrowRatePerBlock	external	Passed	No Issue
10	supplyRatePerBlock	external	Passed	No Issue
11	totalBorrowsCurrent	external	Passed	No Issue
12	borrowBalanceCurrent	external	Passed	No Issue
13	borrowBalanceStored	external	Passed	No Issue
14	exchangeRateCurrent	write	Passed	No Issue
15	exchangeRateStored	read	Passed	No Issue
16	getCash	external	Passed	No Issue
17	accrueInterest	write	Passed	No Issue
18	_setPendingAdmin	external	Passed	No Issue
19	_acceptAdmin	external	Passed	No Issue
20	_setComptroller	write	Passed	No Issue
21	_setReserveFactor	external	Passed	No Issue
22	_reduceReserves	external	Passed	No Issue
23	_setInterestRateModel	write	Passed	No Issue

GovernorAlpha.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	quorumVotes	write	Passed	No Issue
3	proposalThreshold	write	Passed	No Issue
4	proposalMaxOperations	write	Passed	No Issue
5	votingDelay	write	Passed	No Issue
6	votingPeriod	write	Passed	No Issue
7	propose	write	Passed	No Issue
8	queue	write	Passed	No Issue
9	_queueOrRevert	internal	Passed	No Issue
10	execute	write	Passed	No Issue

11	cancel	write	Passed	No Issue
12	getActions	write	Passed	No Issue
13	getReceipt	read	Passed	No Issue
14	state	read	Passed	No Issue
15	castVote	write	Passed	No Issue
16	castVoteBySig	write	Passed	No Issue
17	_castVote	internal	Passed	No Issue
18	__acceptAdmin	write	Passed	No Issue
19	__abdicate	write	Passed	No Issue
20	__queueSetTimelockPendingAdmin	write	Passed	No Issue
21	__executeSetTimelockPendingAdmin	write	Passed	No Issue
22	add256	internal	Passed	No Issue
23	sub256	internal	Passed	No Issue
24	getChainId	internal	Passed	No Issue

LOAN.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	allowance	internal	Passed	No Issue
3	approve	external	Passed	No Issue
4	balanceOf	external	Passed	No Issue
5	transfer	external	Passed	No Issue
6	transferFrom	external	Passed	No Issue
7	delegate	write	Passed	No Issue
8	delegateBySig	write	Passed	No Issue
9	getCurrentVotes	external	Passed	No Issue
10	getPriorVotes	read	Passed	No Issue
11	_delegate	internal	Passed	No Issue
12	_transferTokens	internal	Passed	No Issue
13	_moveDelegates	internal	Passed	No Issue
14	_writeCheckpoint	internal	Passed	No Issue
15	safe32	internal	Passed	No Issue
16	safe96	internal	Passed	No Issue
17	add96	internal	Passed	No Issue
18	sub96	internal	Passed	No Issue
19	getChainId	internal	Passed	No Issue

LoanLens.sol

Functions

Sl.	Functions	Type	Observation	Conclusion

1	constructor	write	Passed	No Issue
2	yTokenMetadata	write	Passed	No Issue
3	yTokenMetadataAll	external	Passed	No Issue
4	yTokenBalances	write	Passed	No Issue
5	yTokenBalancesAll	external	Passed	No Issue
6	yTokenUnderlyingPrice	write	Passed	No Issue
7	yTokenUnderlyingPriceAll	external	Passed	No Issue
8	getAccountLimits	write	Passed	No Issue
9	getGovReceipts	read	Passed	No Issue
10	setProposal	internal	Passed	No Issue
11	getGovProposals	external	Passed	No Issue
12	getLoanBalanceMetadata	external	Passed	No Issue
13	getLoanBalanceMetadataExt	external	Passed	No Issue
14	getLoanVotes	external	Passed	No Issue
15	compareStrings	internal	Passed	No Issue
16	add	internal	Passed	No Issue
17	sub	internal	Passed	No Issue

LoanReserve.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	setLoan	external	Passed	No Issue
3	setRewarder	external	Passed	No Issue
4	setPool	external	access only Owner	No Issue
5	removePool	external	access only Owner	No Issue
6	transfer	external	Passed	No Issue

LoanStaking.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initialize	external	Passed	No Issue
3	addReward	write	access only Owner	No Issue
4	approveRewardDistributor	external	access only Owner	No Issue
5	_rewardPerToken	internal	Passed	No Issue
6	_earned	internal	Passed	No Issue
7	lastTimeRewardApplicable	read	Passed	No Issue
8	rewardPerToken	external	Passed	No Issue
9	getRewardForDuration	external	Passed	No Issue

10	claimableRewards	external	Passed	No Issue
11	totalBalance	external	Passed	No Issue
12	unlockedBalance	external	Passed	No Issue
13	earnedBalances	external	Passed	No Issue
14	lockedBalances	external	Passed	No Issue
15	withdrawableBalance	read	Passed	No Issue
16	stake	external	Passed	No Issue
17	mint	external	Passed	No Issue
18	withdraw	write	Passed	No Issue
19	getReward	write	Passed	No Issue
20	emergencyWithdraw	external	Passed	No Issue
21	withdrawExpiredLocks	external	Passed	No Issue
22	_notifyReward	internal	Passed	No Issue
23	notifyRewardAmount	external	Passed	No Issue
24	recoverERC20	external	access only Owner	No Issue
25	updateReward	modifier	Passed	No Issue
26	getBoost	read	Passed	No Issue
27	getSlots	read	Passed	No Issue
28	getTokenIds	read	Passed	No Issue
29	depositNFT	write	Passed	No Issue
30	withdrawNFT	write	Passed	No Issue
31	setNftController	write	access only Owner	No Issue
32	setNftBoostRate	write	access only Owner	No Issue

NFTController.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getBoostRate	external	Passed	No Issue
3	setWhitelist	external	access only Owner	No Issue
4	setDefaultBoostRate	external	access only Owner	No Issue
5	setBoostRate	external	access only Owner	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens loss
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) High consumption 'for/while' loop :- **Comptroller.sol**

```
function enterMarkets(address[] memory yTokens) public returns (uint[] memory) {
    uint len = yTokens.length;

    uint[] memory results = new uint[](len);
    for (uint i = 0; i < len; i++) {
        YToken yToken = YToken(yTokens[i]);

        results[i] = uint(addToMarketInternal(yToken, msg.sender));
    }

    return results;
}
```

As seen in the AS-IS section, there are several places in the smart contracts, where array.length is used directly in the loops. It is recommended to put some kind of limits, so it does not go wild and create any scenario where it can hit the block gas limit.

Resolution: We suggest using some limits for the array.

(2) Loan Rate has no limit: **Comptroller.sol**

```
function _setLoanRate(uint loanRate_) public {
    require(adminOrInitializing(), "only admin can change loan rate");

    uint oldRate = loanRate;
    loanRate = loanRate_;
    emit NewLoanRate(oldRate, loanRate_);

    refreshLoanSpeeds();
```

Admin can set the loan rate to any variable. This might deter investors as they could be wary that this rate might one day be set to 100% to force transfers to go to the contract admin role.

Resolution: Consider adding an explicit limit while setting the loan rate value.

Very Low / Informational / Best practices

(1) Use the latest solidity version while contract deployment to prevent any compiler version level bugs.

(2) Use visibility External over public: If any function is not being called internally, then it is better to specify its visibility as external. It saves some gas as well. https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices

(3) Division before multiplication: **DAIInterestRateModelV3.sol**

```
 */
function poke() public {
    (uint duty, ) = jug.ilks("ETH-A");
    uint stabilityFeePerBlock = duty.add(jug.base()).sub(1e27).mul(1e18).div(1e27).mul(15);

    // We ensure the minimum borrow rate >= DSR / (1 - reserve factor)
    baseRatePerBlock = dsrPerBlock().mul(1e18).div(assumedOneMinusReserveFactorMantissa);

    // The roof borrow rate is max(base rate, stability fee) + gap, from which we derive the slope
    if (baseRatePerBlock < stabilityFeePerBlock) {
        multiplierPerBlock = stabilityFeePerBlock.sub(baseRatePerBlock).add(gapPerBlock).mul(1e18).div(kink);
    } else {
        multiplierPerBlock = gapPerBlock.mul(1e18).div(kink);
    }

    emit NewInterestParams(baseRatePerBlock, multiplierPerBlock, jumpMultiplierPerBlock, kink);
}
```

Solidity being resource constraint language, dividing any amount and then multiplying will cause discrepancy in the outcome. Therefore always multiply the amount first and then divide it.

Resolution: Consider ordering multiplication before division.

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- __executeSetTimelockPendingAdmin: GovernorAlpha owner can execute pending timelock.
- __queueSetTimelockPendingAdmin: GovernorAlpha owner can set queue pending timelock.
- __abdicate: GovernorAlpha owner can execute abdicate function.
- __acceptAdmin: GovernorAlpha owner can accept admin.
- _setPriceOracle: Comptroller admin can set a new price oracle for the comptroller.
- _setCloseFactor: Comptroller admin can set the closeFactor used when liquidating borrows.
- _setCollateralFactor: Comptroller admin can set the collateralFactor for a market.
- _setMaxAssets: Comptroller admin can set a maxAssets value.
- _setLiquidationIncentive: Comptroller admin can set liquidation Incentive value.
- _supportMarket: Comptroller admin can set isListed and add support for the market token.
- _setPauseGuardian: Comptroller admin can change the Pause Guardian address.
- _setMintPaused: Comptroller admin can set mint pause status.
- _setBorrowPaused: Comptroller admin can set borrowed paused status.
- _setTransferPaused: Comptroller admin can set transfer paused status.
- _setSeizePaused: Comptroller admin can set seized status.
- _setReserveInfo: Comptroller admin can set reserve information.
- _become: Comptroller admin can change new brains.
- adminOrInitializing: Comptroller admin can check admin, or this contract is becoming the new implementation.
- _dropLoanMarket: Comptroller admin can remove a market from loanMarkets, preventing it from earning LOAN in the flywheel yToken. The address of the market to drop.
- _setCollateralFactor: ComptrollerG1 admin can set per-market collateralFactor.
- _setMaxAssets: ComptrollerG1 admin can set maxAssets.
- _setLiquidationIncentive: ComptrollerG1 admin can set liquidation incentive.
- _supportMarket: ComptrollerG1 admin can set isListed and add support for the market tokon.
- _become: ComptrollerG1 admin can change brains.
- _setPriceOracle: ComptrollerG2 admin can set a new price oracle for the comptroller.

- _setCloseFactor: ComptrollerG2 admin can set the closeFactor used when liquidating borrows.
- _setCollateralFactor: ComptrollerG2 admin can set the collateralFactor for a market.
- _setMaxAssets: ComptrollerG2 admin can set a maxAssets value.
- _setLiquidationIncentive: ComptrollerG2 admin can set liquidation Incentive value.
- _supportMarket: ComptrollerG2 admin can set isListed and add support for the market token.
- _setPauseGuardian: ComptrollerG2 admin can set pause guardian status.
- _setMintPaused: ComptrollerG2 admin can set mint pauses status.
- _setBorrowPaused: ComptrollerG2 admin can set borrowed paused status.
- _setTransferPaused: ComptrollerG2 admin can set transfer paused status.
- _setSeizePaused: ComptrollerG2 admin can set seized status.
- _become: ComptrollerG2 admin can change new brains.
- _setPriceOracle: ComptrollerG3 admin can set a new price oracle for the comptroller.
- setPriceOracle: ComptrollerG3 admin can set a new price oracle for the comptroller.
- _setCloseFactor: ComptrollerG3 admin can set the closeFactor used when liquidating borrows.
- _setCollateralFactor: ComptrollerG3 admin can set the collateralFactor for a market.
- _setMaxAssets: ComptrollerG3 admin can set a maxAssets value.
- _setLiquidationIncentive: ComptrollerG3 admin can set liquidation Incentive value.
- _supportMarket: ComptrollerG3 admin can set isListed and add support for the market token.
- _setPauseGuardian: ComptrollerG3 admin can change the Pause Guardian address.
- _setMintPaused: ComptrollerG3 admin can set mint pause status.
- _setBorrowPaused: ComptrollerG3 admin can set borrowed paused status.
- _setTransferPaused: ComptrollerG3 admin can set transfer paused status.
- _setSeizePaused: ComptrollerG3 admin can set seized status.
- _become: ComptrollerG3 admin can change new brains.
- _becomeG3: ComptrollerG3 admin can change new brains.

- adminOrInitializing: ComptrollerG3 admin can check the caller is admin, or this contract is becoming the new implementation.
- _setLoanRate: ComptrollerG3 admin can set the amount of LOAN distributed per block.
- _addLoanMarkets: ComptrollerG3 admin can add markers to loanMarkets, allowing them to earn LOAN in the flywheel yToken.
- _dropLoanMarket: ComptrollerG3 admin can remove a market from loanMarkets, preventing it from earning LOAN in the flywheel yToken.
- _setPriceOracle: ComptrollerG4 admin can set a new price oracle for the comptroller.
- _setCloseFactor: ComptrollerG4 admin can set the closeFactor used when liquidating borrows.
- _setCollateralFactor: ComptrollerG4 admin can set the collateralFactor for a market.
- _setMaxAssets: ComptrollerG4 admin can set a maxAssets value.
- _setLiquidationIncentive: ComptrollerG4 admin can set liquidation Incentive value.
- _supportMarket: ComptrollerG4 admin can set isListed and add support for the market token.
- _setPauseGuardian: ComptrollerG4 admin can change the Pause Guardian address.
- _setMintPaused: ComptrollerG4 admin can set mint pause status.
- _setBorrowPaused: ComptrollerG4 admin can set borrowed paused status.
- _setTransferPaused: ComptrollerG4 admin can set transfer paused status.
- _setSeizePaused: ComptrollerG4 admin can set seized status.
- _become: ComptrollerG4 admin can change new brains.
- _setLoanSpeed: ComptrollerG4 admin can set loan speed.
- _setLoanRate:ComptrollerG4 admin can set the amount of LOAN distributed per block.
- _addLoanMarkets: ComptrollerG4 admin can add markers to loanMarkets, allowing them to earn LOAN in the flywheel yToken.
- _dropLoanMarket: ComptrollerG4 admin can remove a market
- _setPriceOracle: ComptrollerG5 admin can set a new price oracle for the comptroller.

- _setCloseFactor: ComptrollerG5 admin can set the closeFactor used when liquidating borrows.
- _setCollateralFactor: ComptrollerG5 admin can set the collateralFactor for a market.
- _setMaxAssets: ComptrollerG5 admin can set a maxAssets value.
- _setLiquidationIncentive: ComptrollerG5 admin can set liquidation Incentive value.
- _supportMarket: ComptrollerG5 admin can set isListed and add support for the market token.
- _setPauseGuardian: ComptrollerG5 admin can change the Pause Guardian address.
- _setMintPaused: ComptrollerG5 admin can set mint pause status.
- _setBorrowPaused: ComptrollerG5 admin can set borrowed paused status.
- _setTransferPaused: ComptrollerG5 admin can set transfer paused status.
- _setSeizePaused: ComptrollerG5 admin can set seized status.
- _become: ComptrollerG5 admin can change new brains.
- adminOrInitializing: ComptrollerG5 admin can check admin, or this contract is becoming the new implementation.
- _setLoanSpeeds: ComptrollerG5 admin can set loan speed.
- _setContributorLoanSpeed: ComptrollerG5 admin can set contributor load speed.
- _grantLOAN: ComptrollerG5 admin can grant loan.
- _setLoanRate: ComptrollerG5 admin can set the amount of LOAN distributed per block.
- _dropLoanMarket: ComptrollerG5 admin can remove a market.
- updateJumpRateModel: DAIInterestRateModelV3 owner can update the parameters of the interest rate model.
- updateJumpRateModel: JumpRateModelV2 owner can update the parameters of the interest rate model.
- setUnderlyingPrice: LoanAggregatorPriceOracle owner can set underlying price.
- setDirectPrice: LoanAggregatorPriceOracle owner can set direct price.
- setFeed: LoanAggregatorPriceOracle owner can set a feed address.
- setRef: LoanAggregatorPriceOracle owner can set a new reference address.
- setAdmin: LoanAggregatorPriceOracle owner can set a new admin.
- setRefAddress: LoanPriceOracle owner can set reference address.
- setUnderlyingPrice: LoanPriceOracle owner can set underlying price.

- setDirectPrice: LoanPriceOracle owner can set direct price.
- setAdmin: LoanPriceOracle owner can set a new admin.
- setDelay: Timelock owner can set delay time.
- acceptAdmin: Timelock owner can accept admin request.
- setPendingAdmin: Timelock owner can set pending admin request.
- queueTransaction: Timelock owner can queue transactions.
- cancelTransaction: Timelock owner can cancel transaction.
- executeTransaction: Timelock owner can execute transaction.
- _setPendingImplementation: Unitroller owner can set pending implementation.
- _setPendingAdmin: Unitroller owner can set pending admin rights.
- _resignImplementation: YDaiDelegate owner can resign implementation.
- _setImplementation: YErc20Delegator owner can update the implementation of the delegator.
- _delegateLoanLikeTo: YLoanLikeDelegate owner can delegate the votes of the underlying LOAN-like underlying.
- _setPendingAdmin: YToken owner can set pending admin rights.
- _acceptAdmin: YToken owner can accept transfer of admin rights.
- _setComptroller: YToken owner can set a new comptroller for the market.
- _setReserveFactorFresh: YToken owner can set a new reserve factor for the protocol.
- __acceptAdmin: GovernorAlpha owner can accept admin request.
- __abdicate: GovernorAlpha owner can abdicate admin.
- __queueSetTimelockPendingAdmin: GovernorAlpha owner can queue set time lock pending admin.
- __executeSetTimelockPendingAdmin: GovernorAlpha owner can execute set pending timelock.
- setPool: LoanReserve owner can set a new pool address.
- removePool: LoanReserve owner can remove pool address.
- transfer: LoanReserve owners can transfer funds and withdraw.
- addReward: LoanStaking owner can add a new reward token to be distributed to stakers.
- approveRewardDistributor: LoanStaking owner can modify approval for an address to call notifyRewardAmount.

- recoverERC20: LoanStaking owner can add to support recovering LP Rewards from other systems such as BAL to be distributed to holders.
- setNftController: LoanStaking owner can set NFT controller address.
- setNftBoostRate: LoanStaking owner can set NFT boost rate.
- setWhitelist: NFTController owner can set whitelist addresses.
- setDefaultBoostRate: NFTController owner can set default boost rate value.
- setBoostRate: NFTController owner can set boost rate value.

To make the smart contract 100% decentralized, we suggest renouncing ownership in the smart contract once its function is completed.

Conclusion

We were given a contract code in the form of files. And we have used all possible tests based on given objects as files. We have not observed any major issues in the smart contracts. **So, the smart contracts are ready for the mainnet deployment**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured"**.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

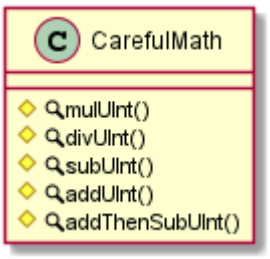
Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

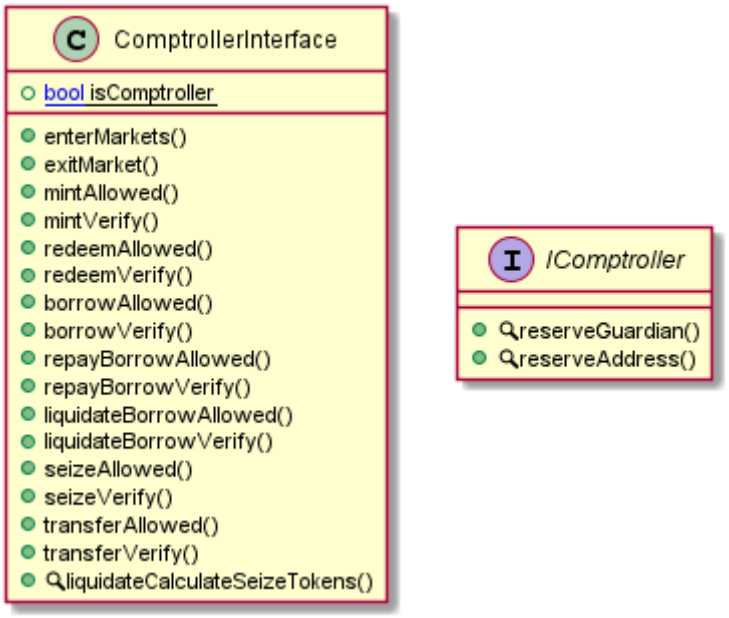
Appendix

Code Flow Diagram - Yield Mountain Protocol

CarefulMath Diagram

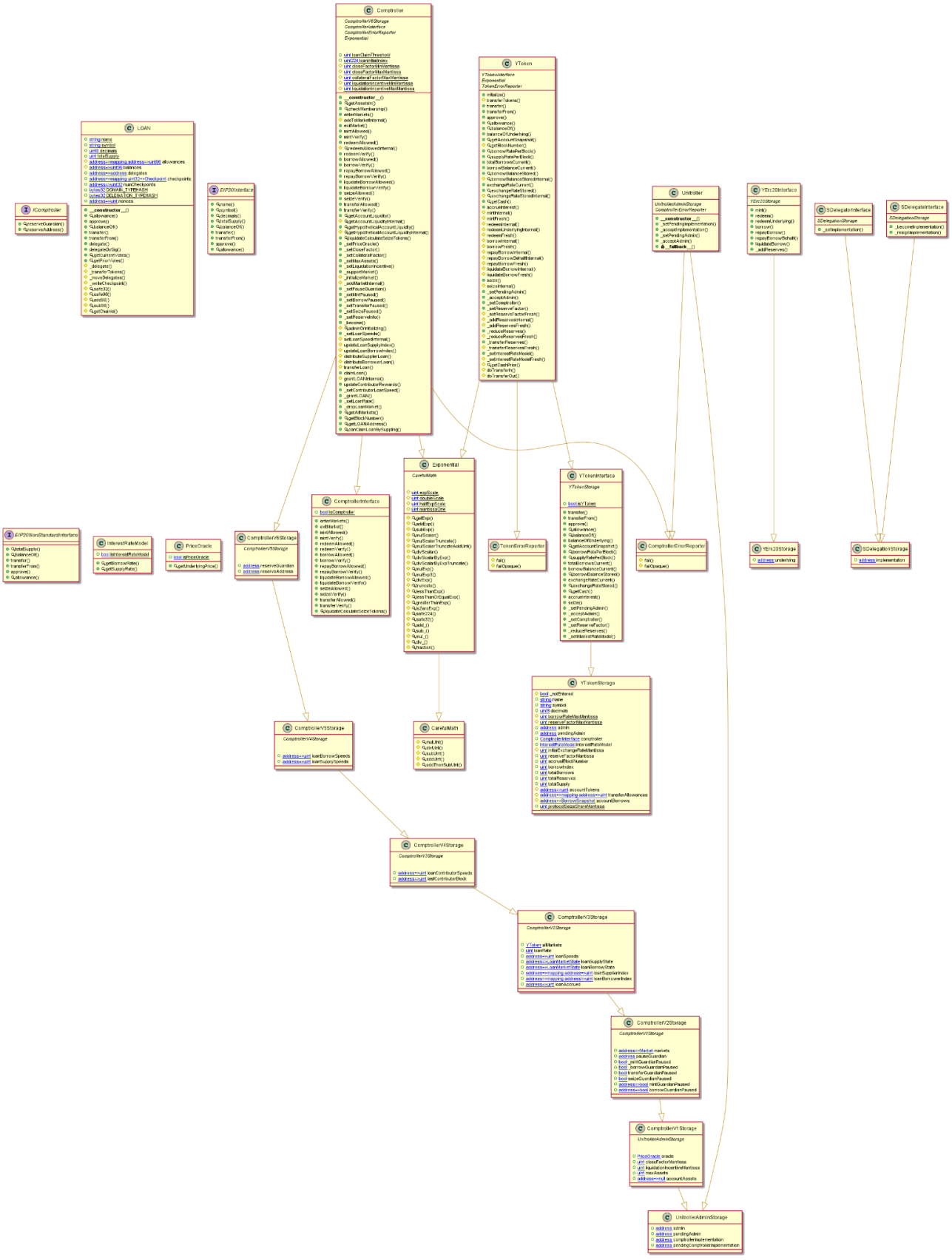


ComptrollerInterface Diagram

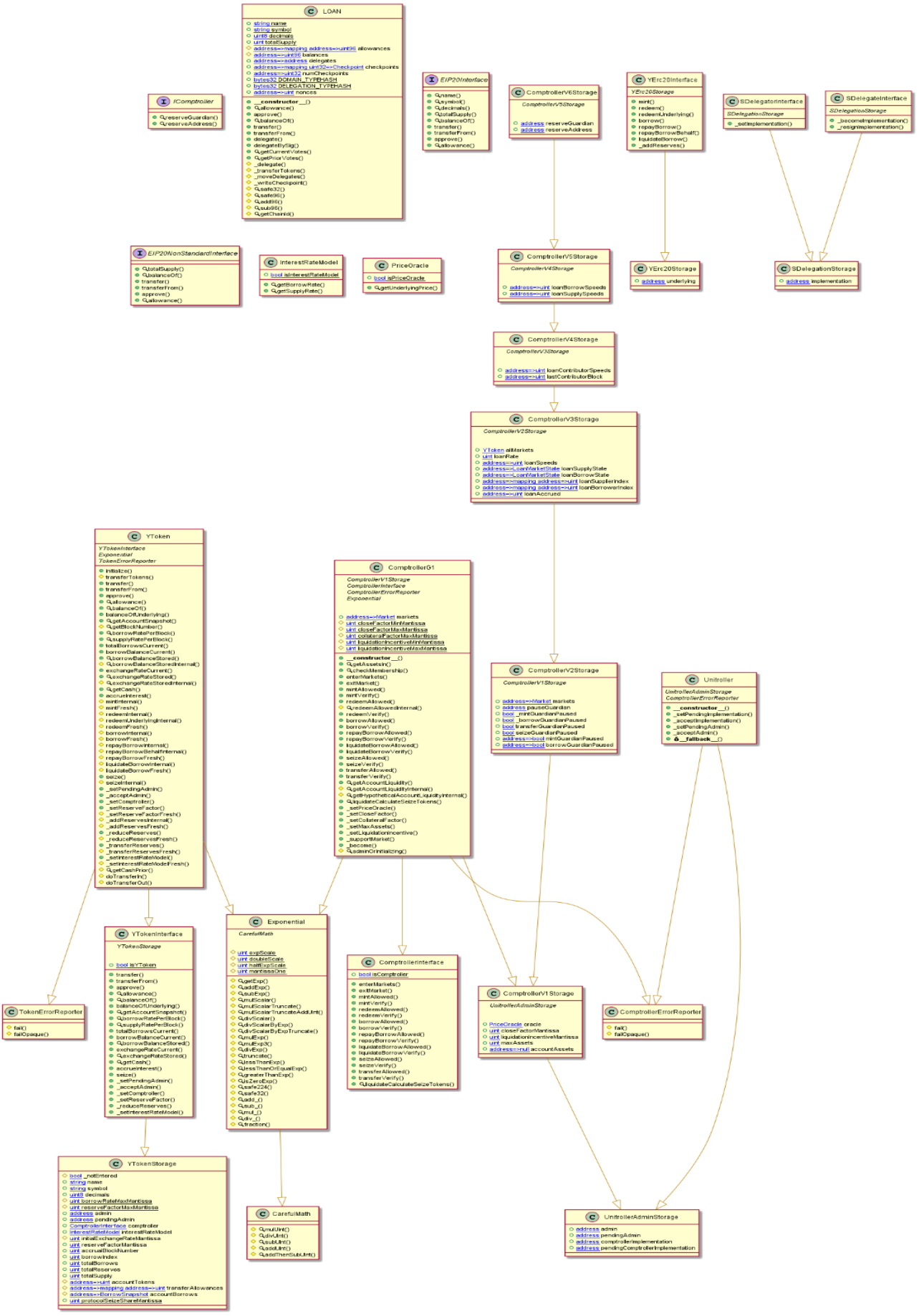


ErrorReporter Diagram

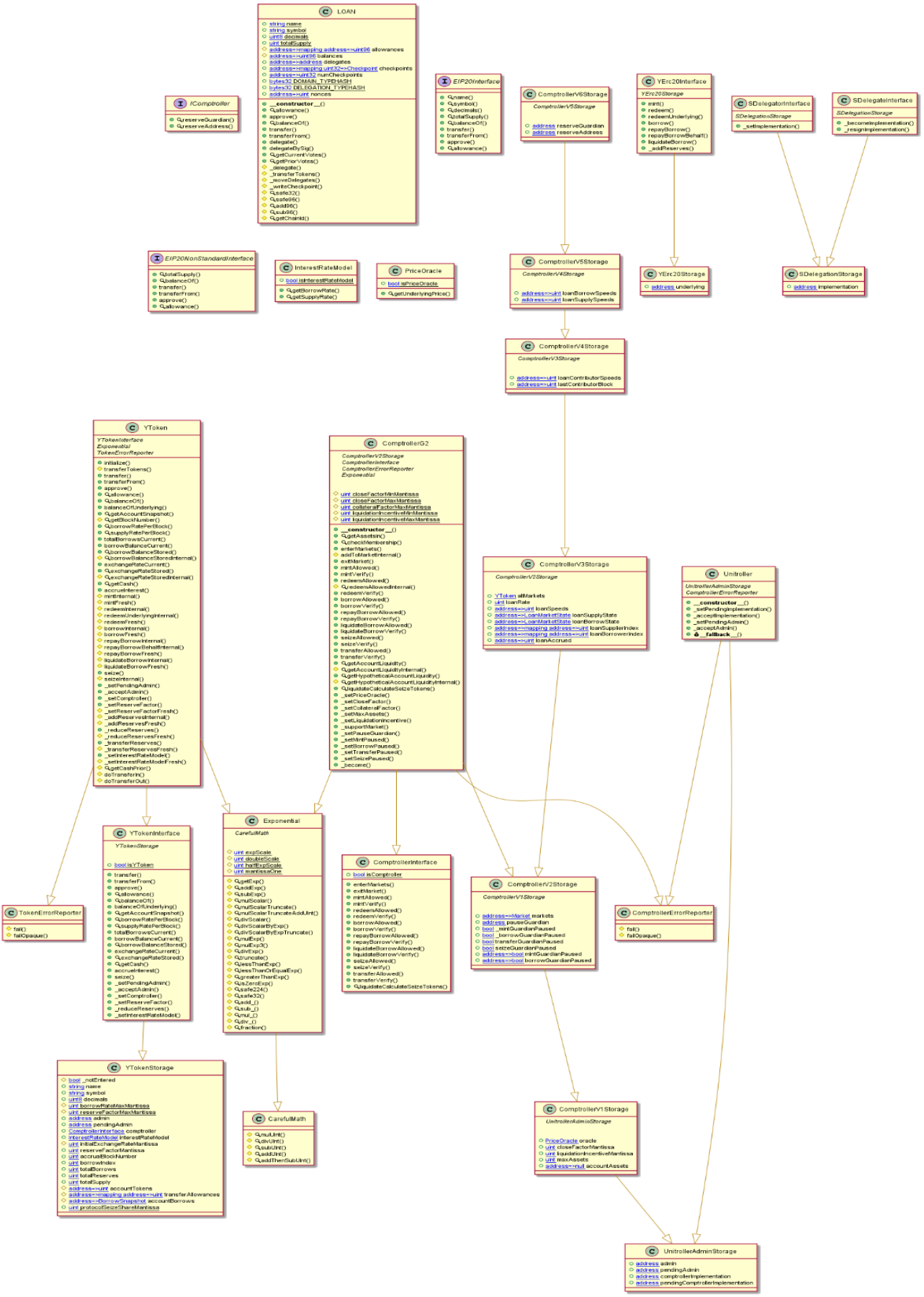
Comptroller Diagram



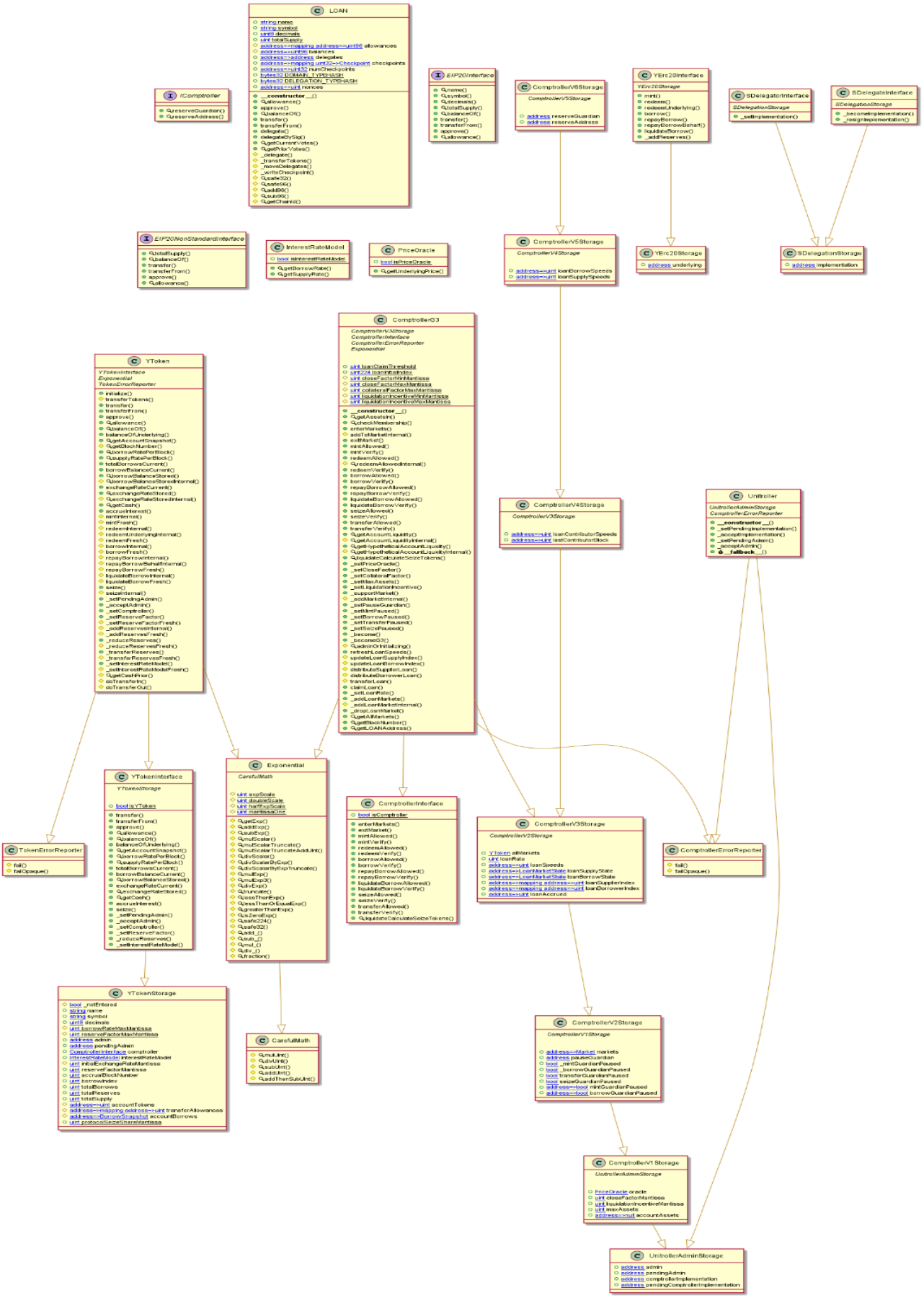
ComptrollerG1 Diagram



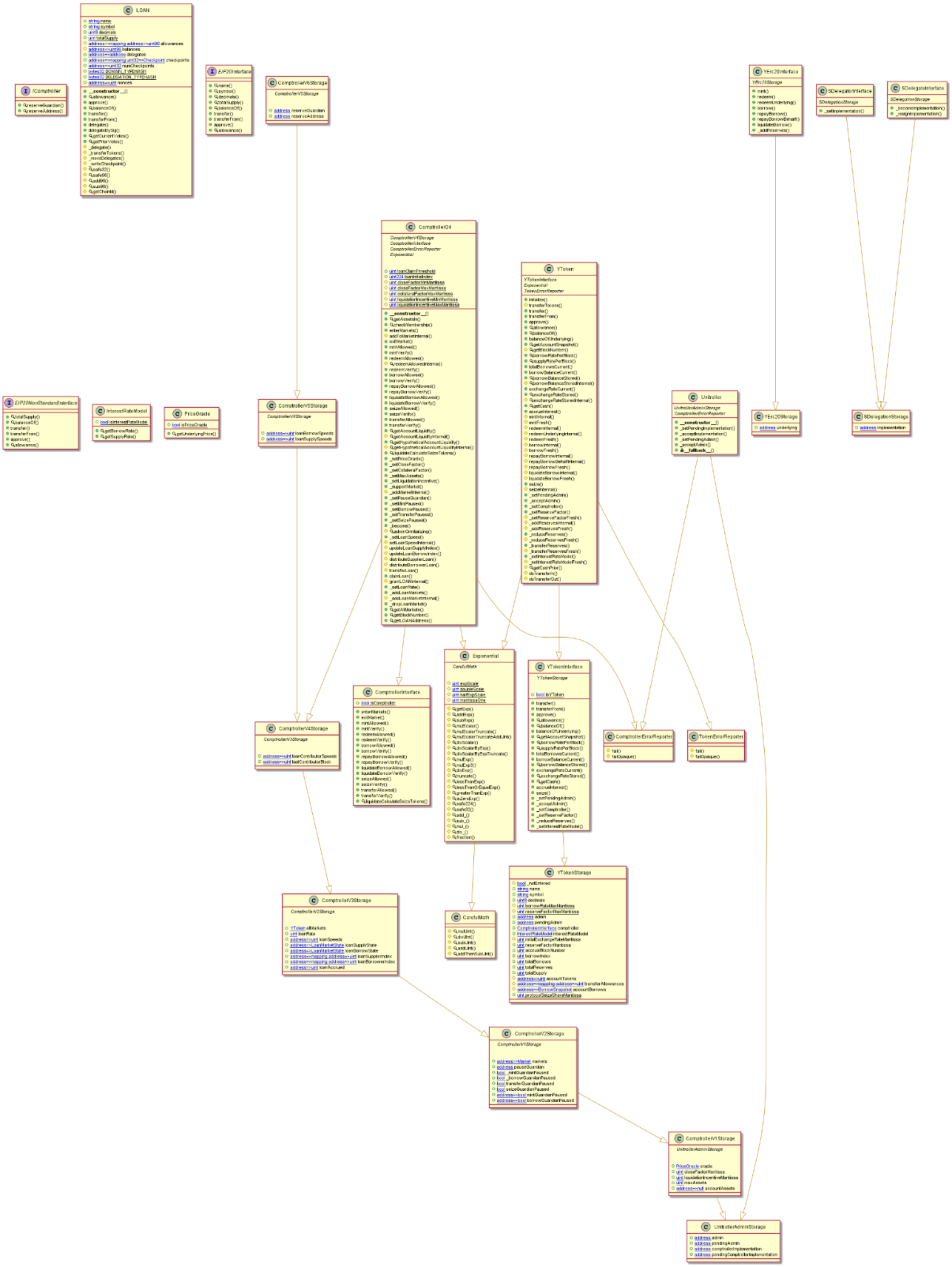
ComptrollerG2 Diagram



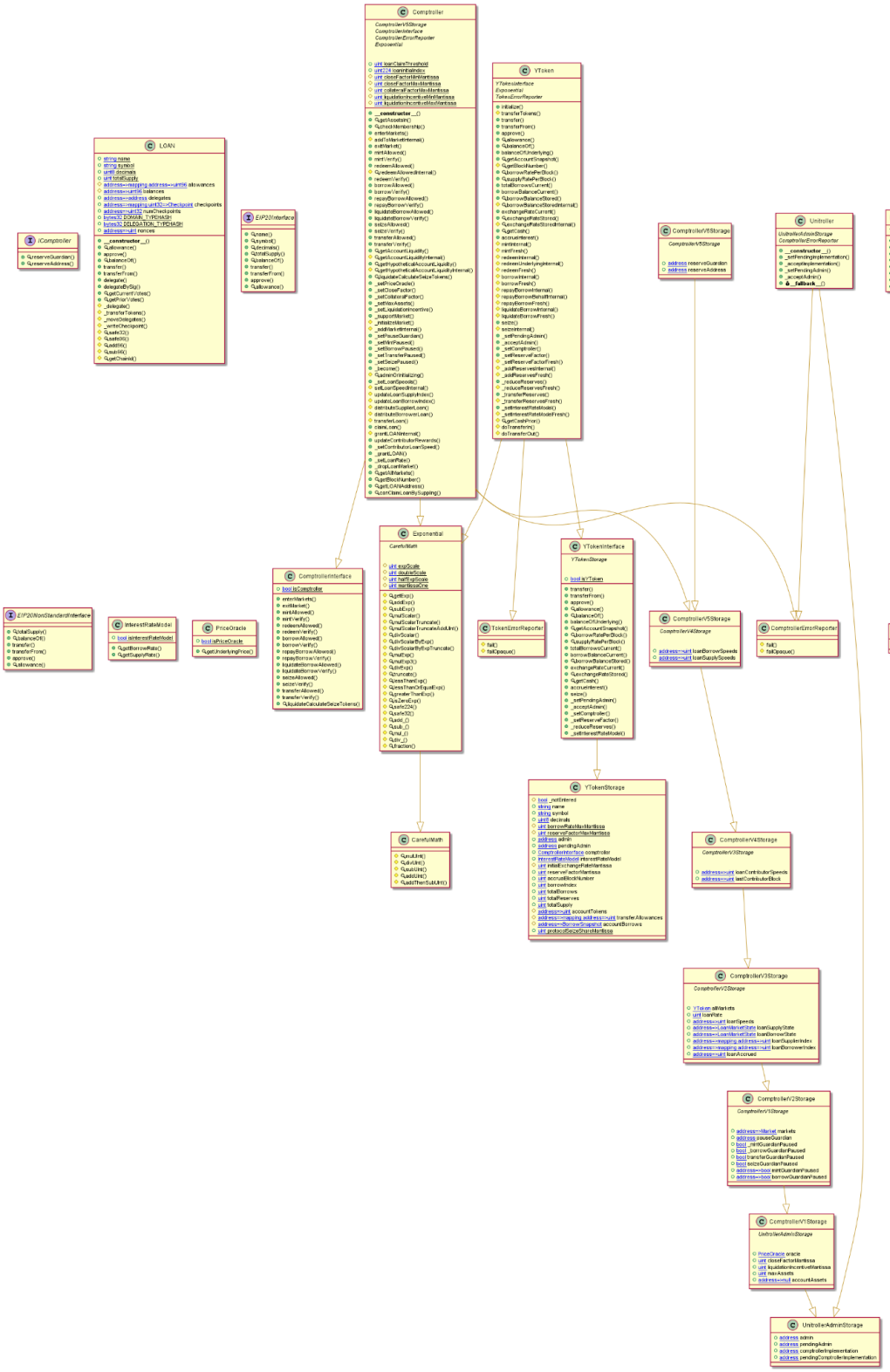
ComptrollerG3 Diagram



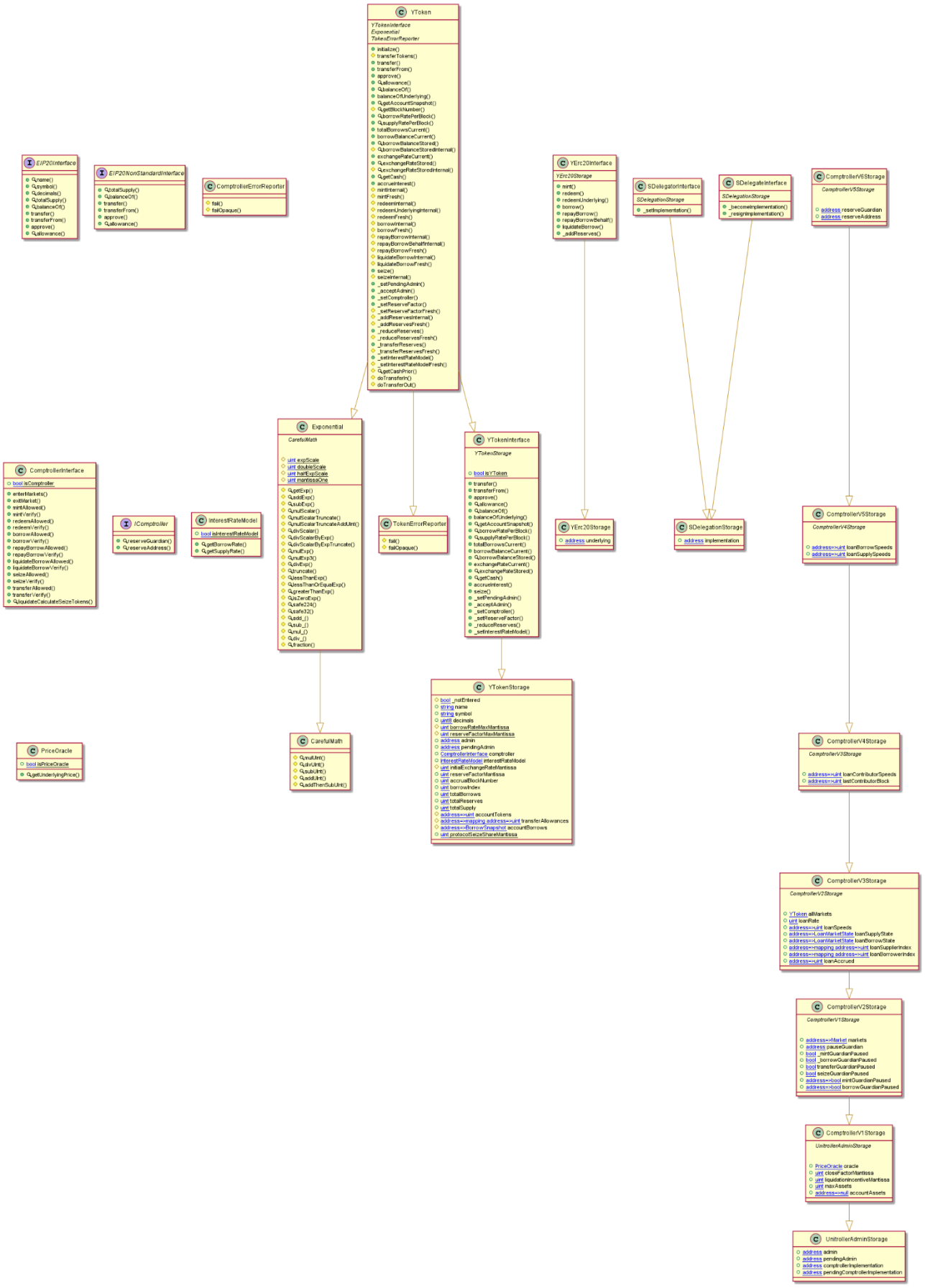
ComptrollerG4 Diagram



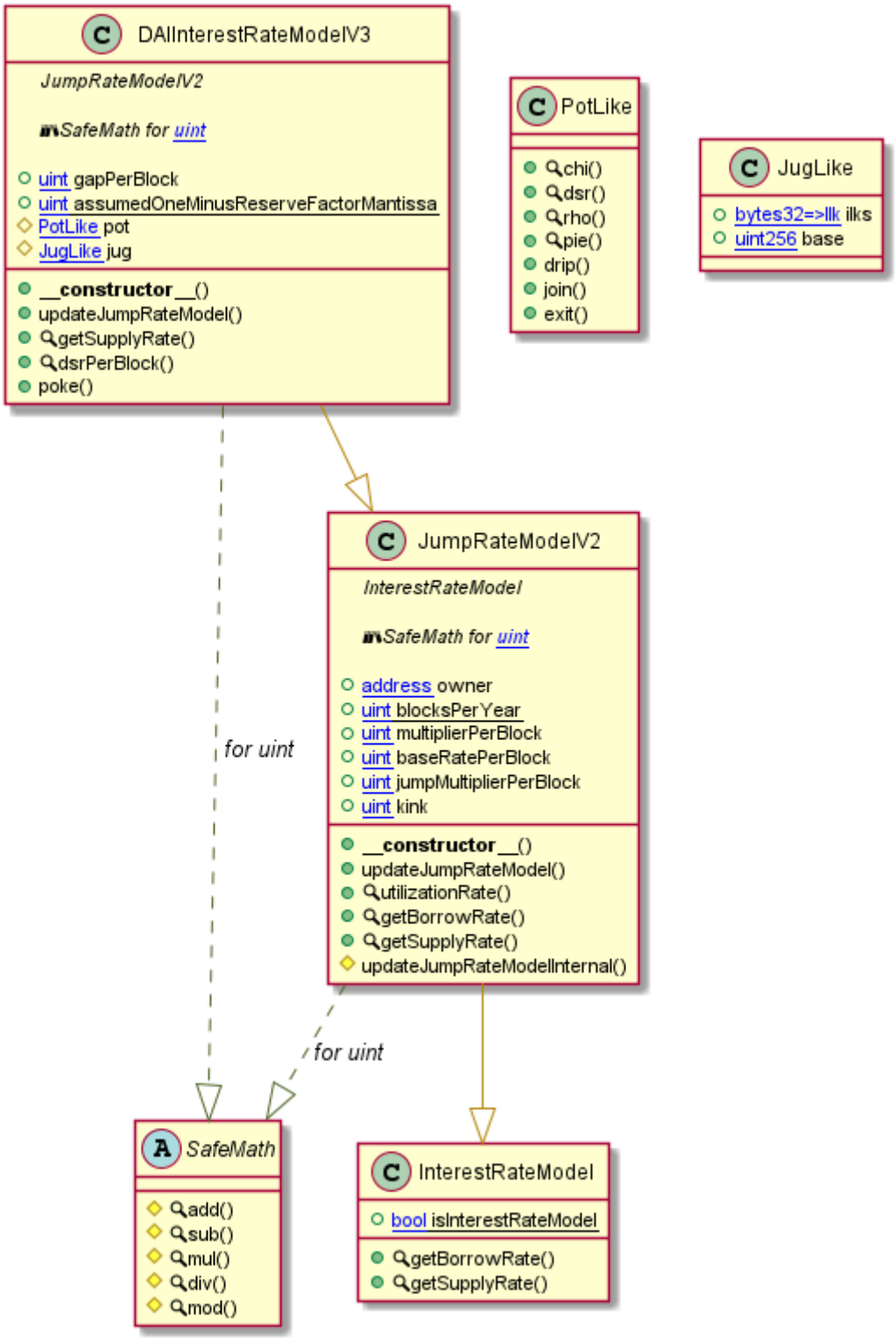
ComptrollerG5 Diagram



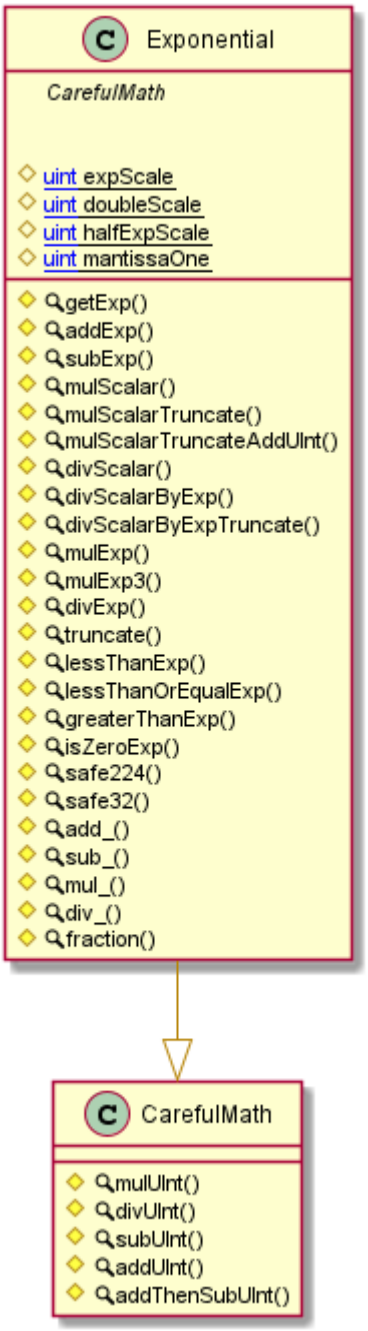
ComptrollerStorage Diagram



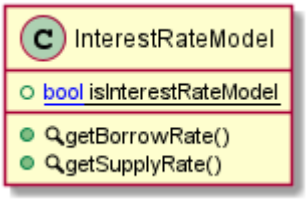
DAIInterestRateModelV3 Diagram



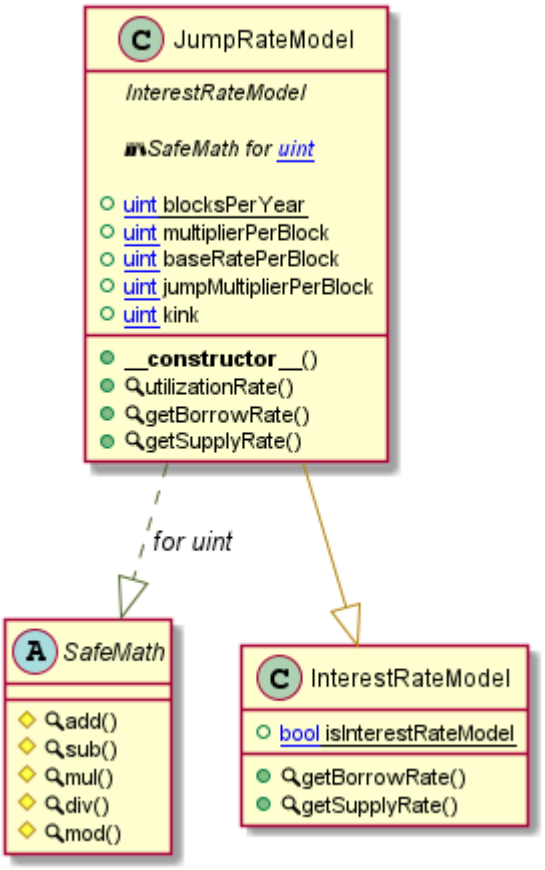
Exponential Diagram



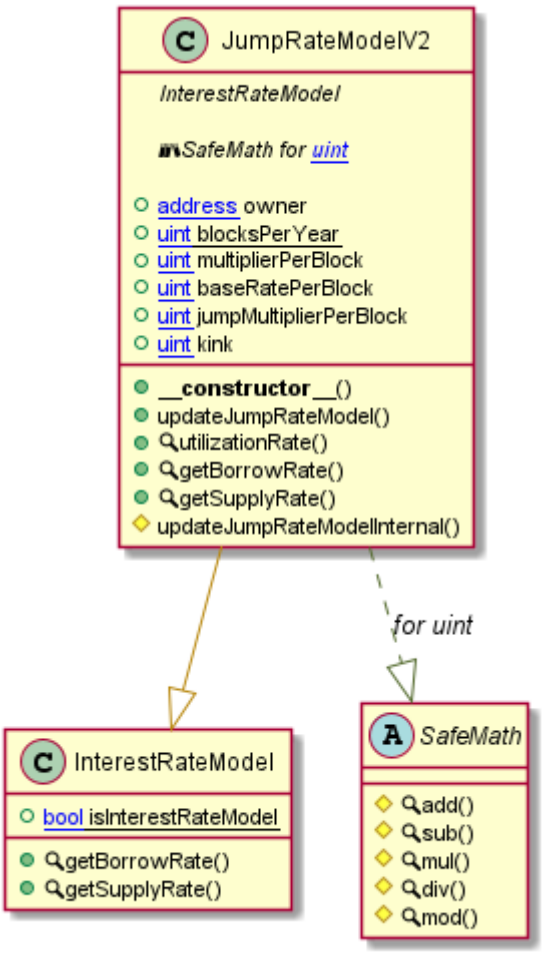
InterestRateModel Diagram



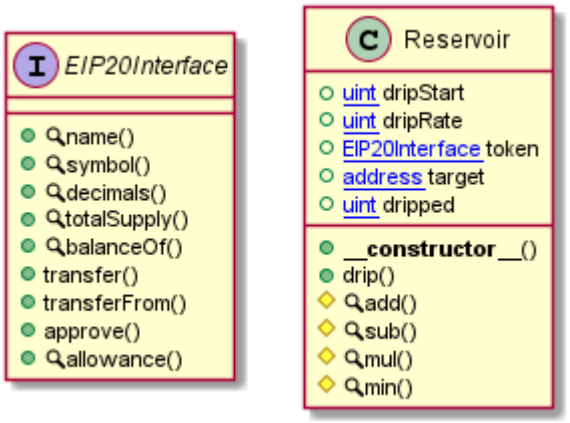
JumpRateModel Diagram



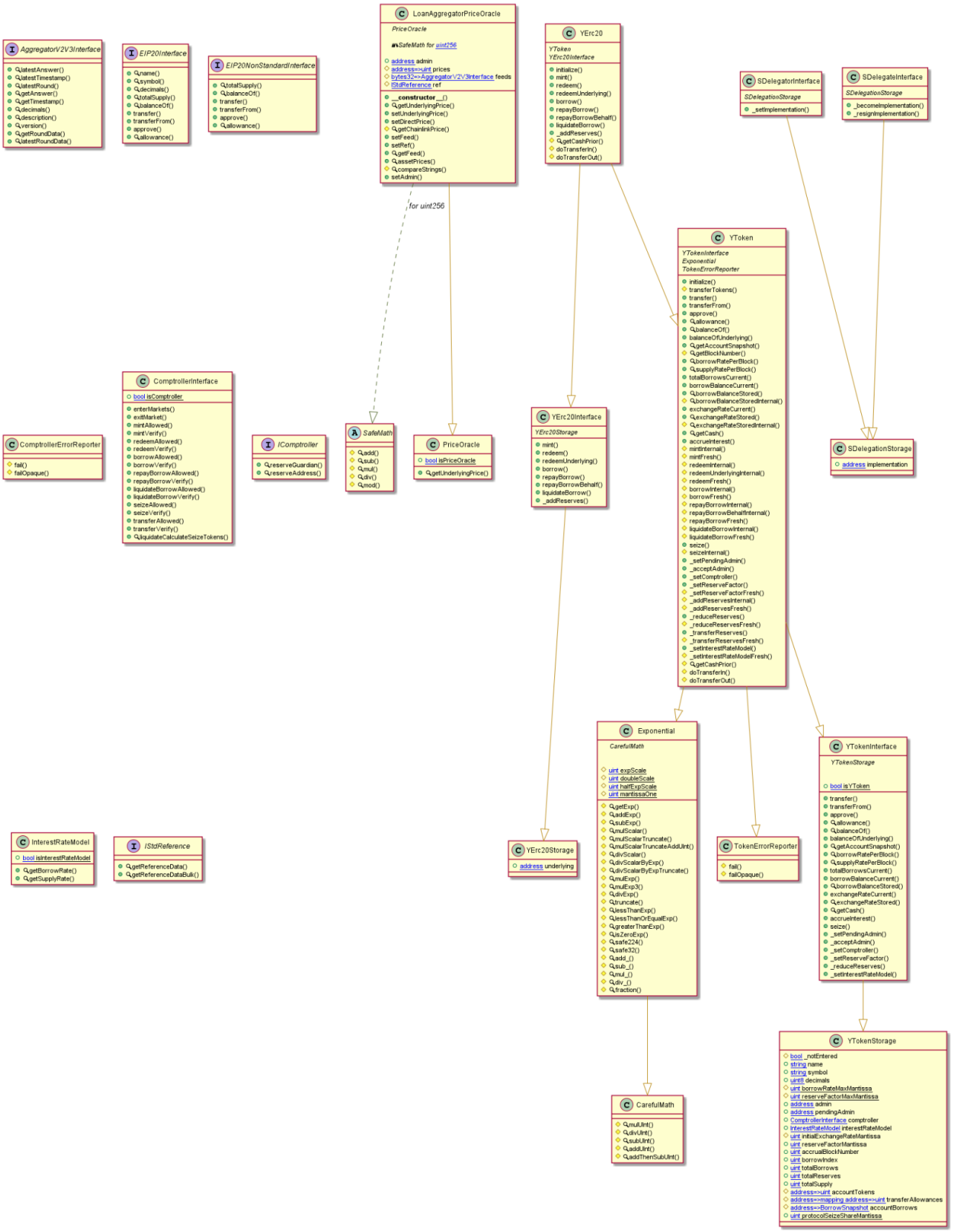
JumpRateModelV2 Diagram



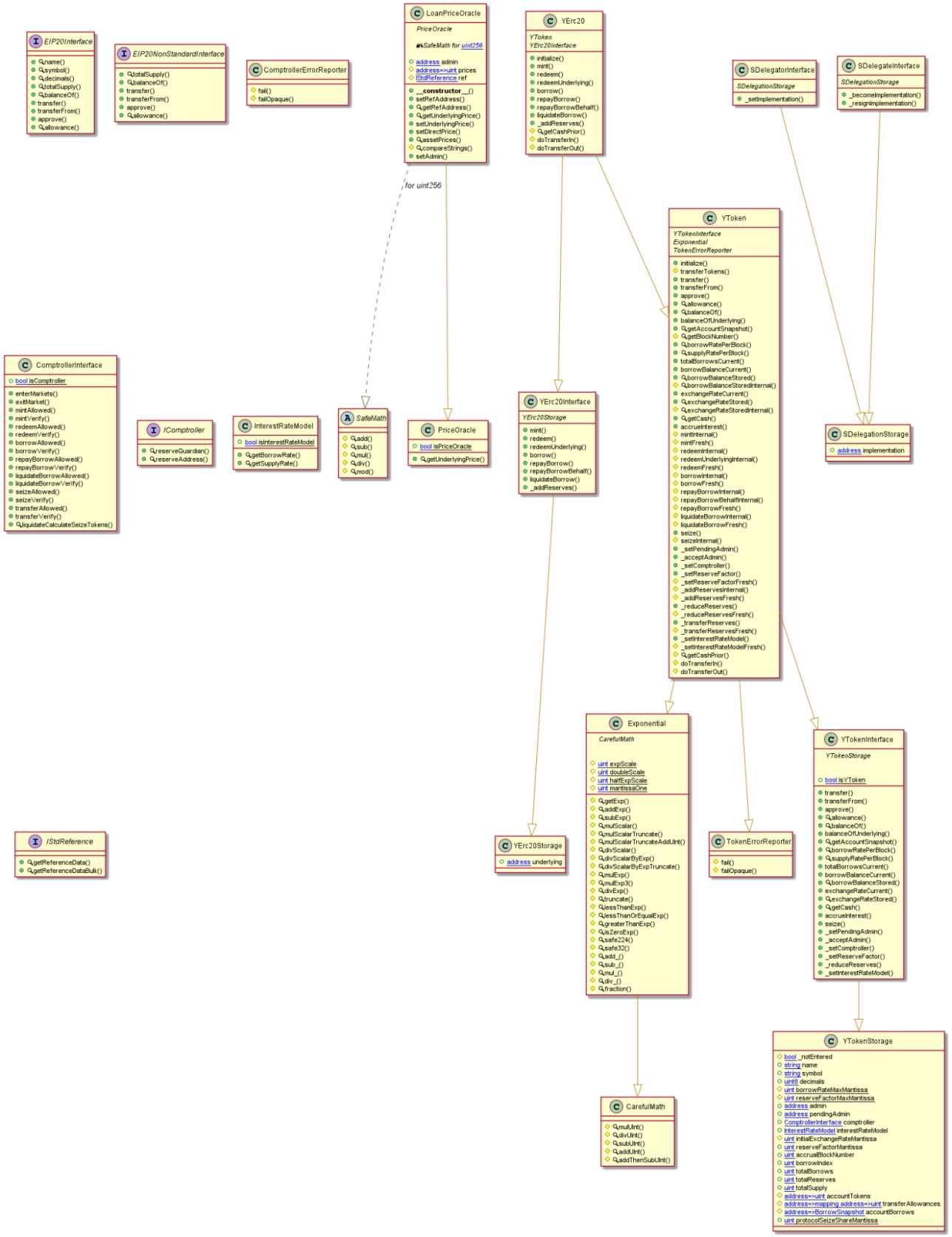
Reservoir Diagram



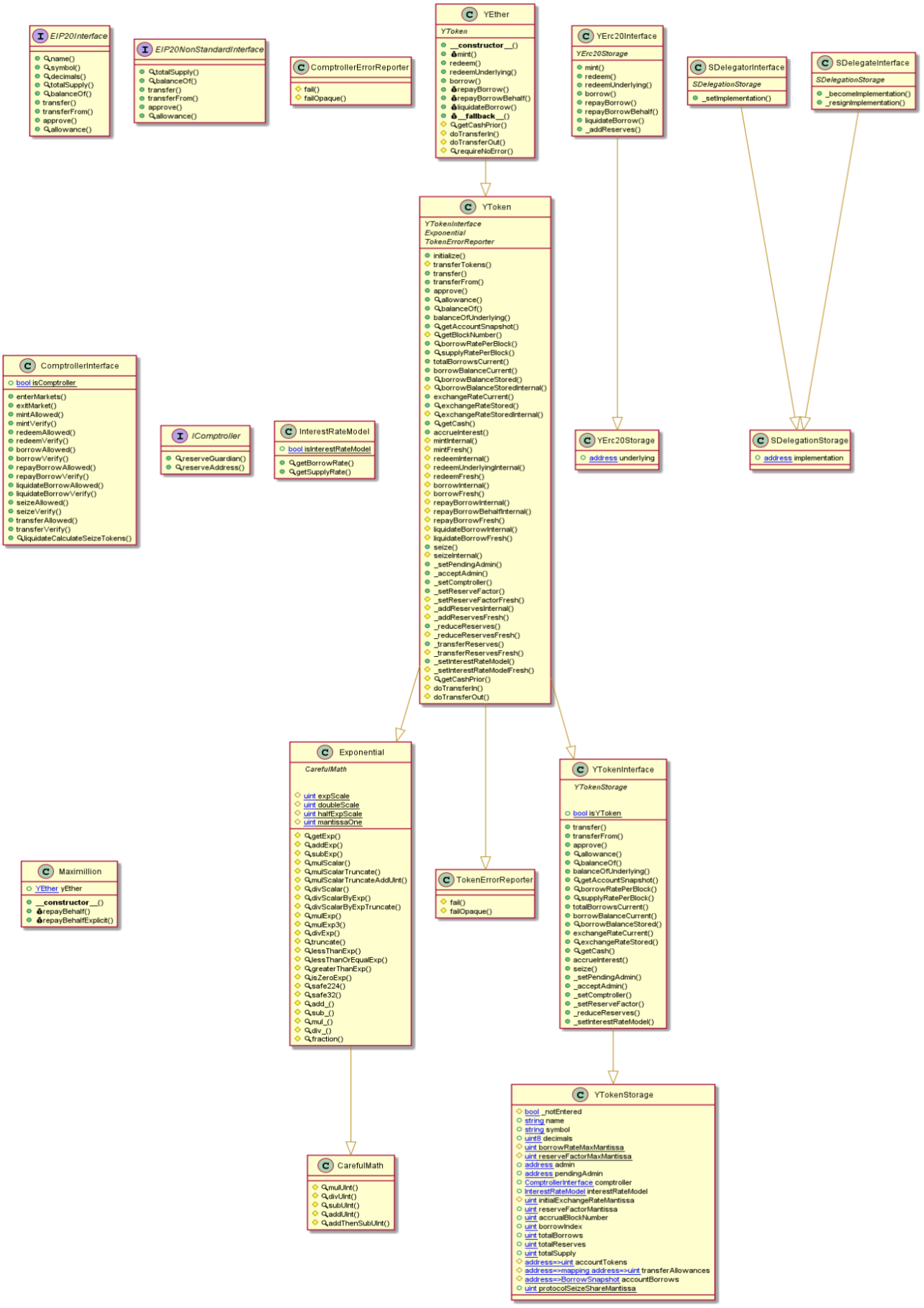
LoanAggregatorPriceOracle Diagram



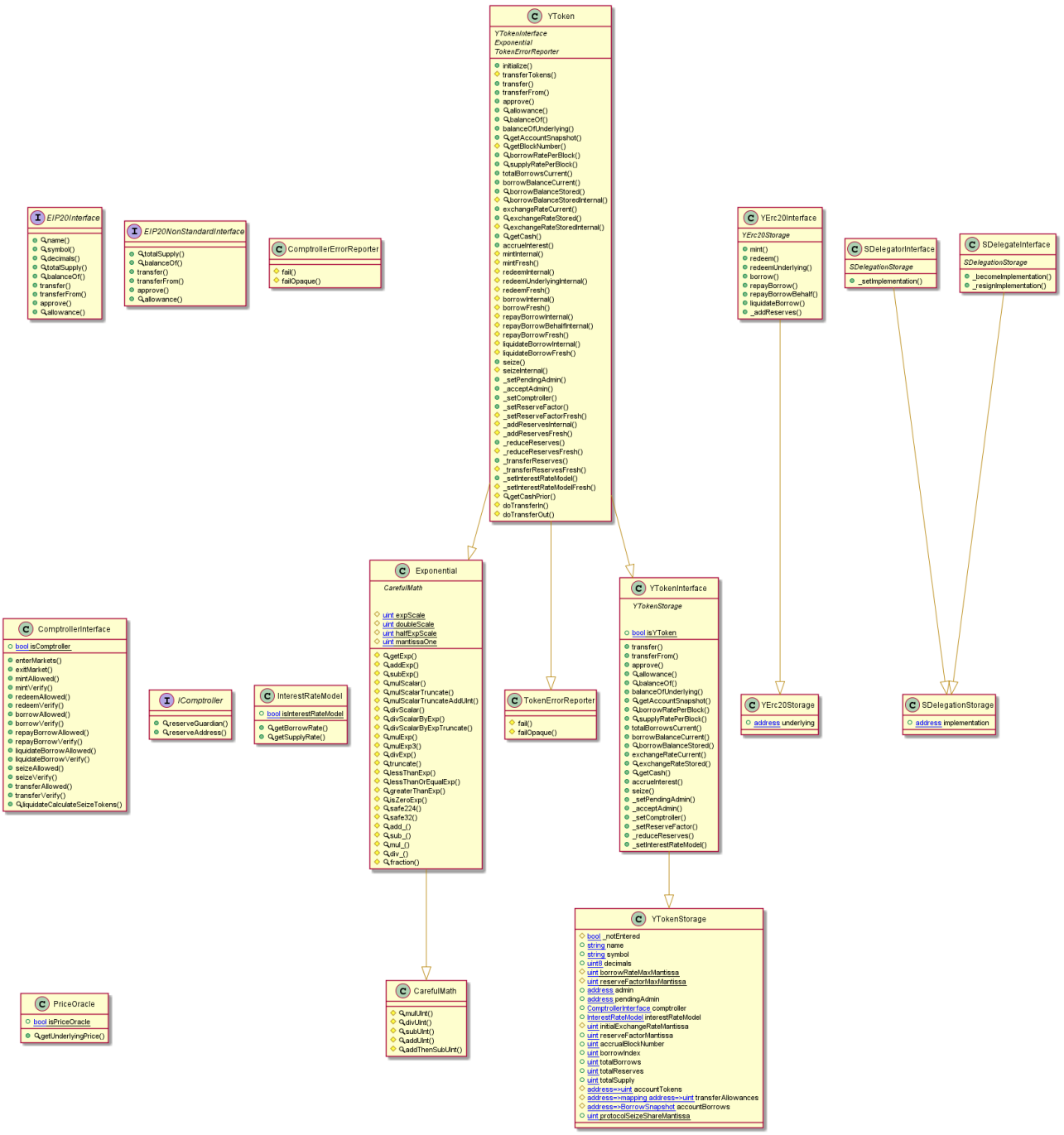
LoanPriceOracle Diagram



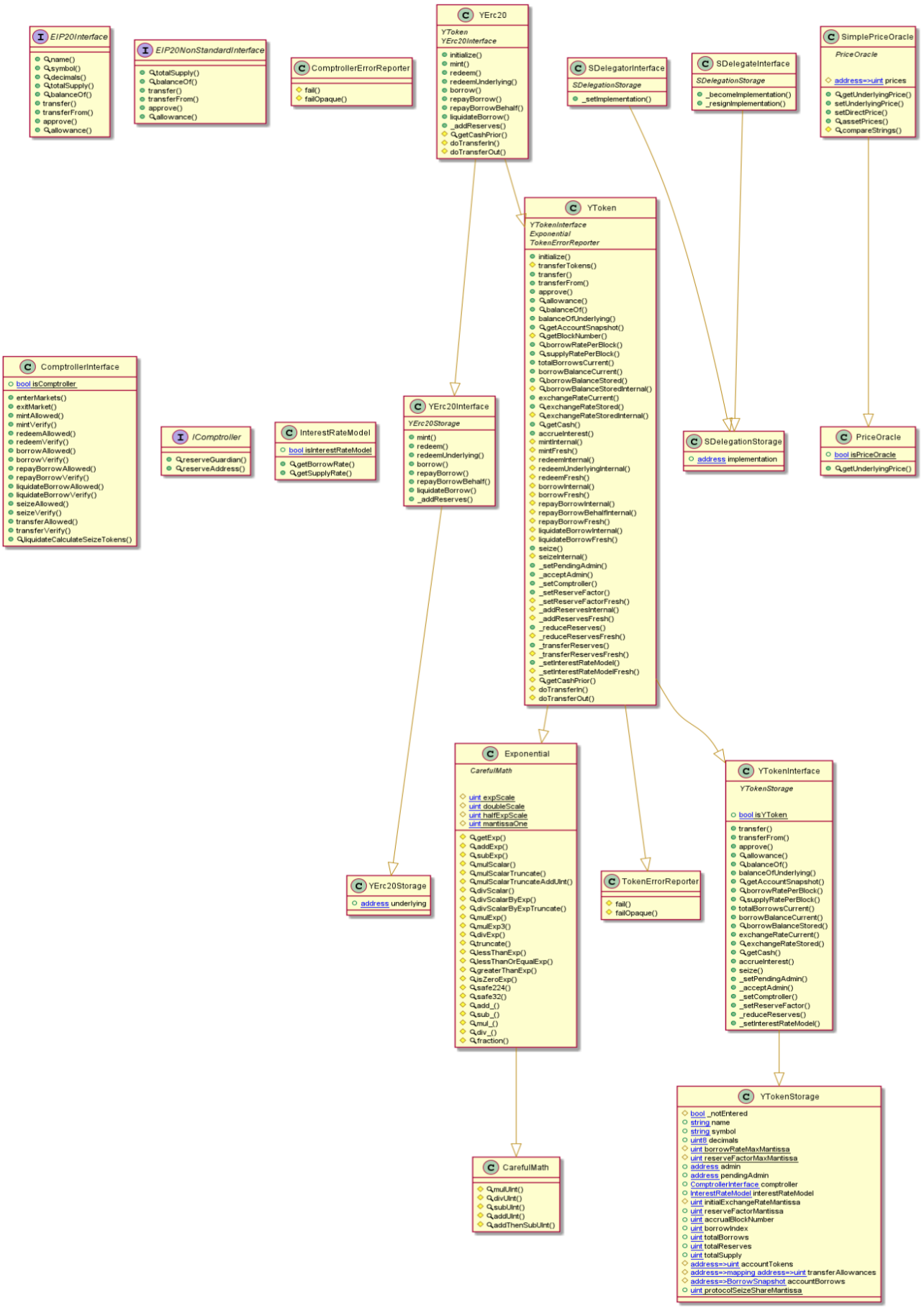
Maximillion Diagram



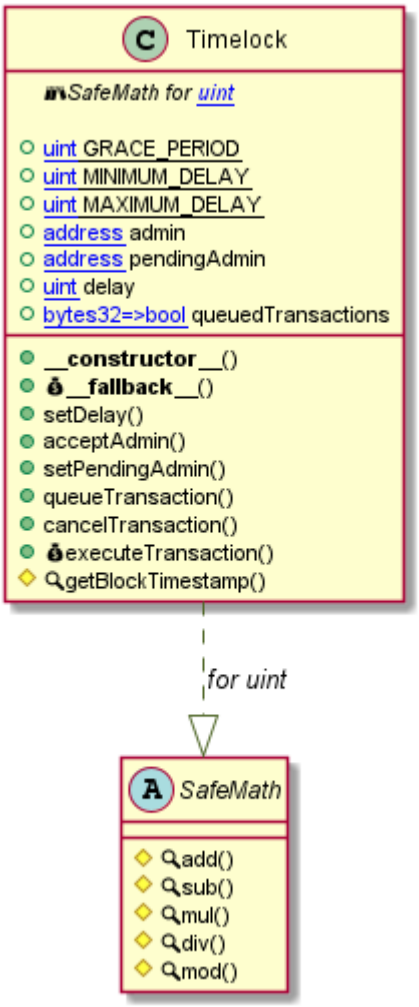
PriceOracle Diagram



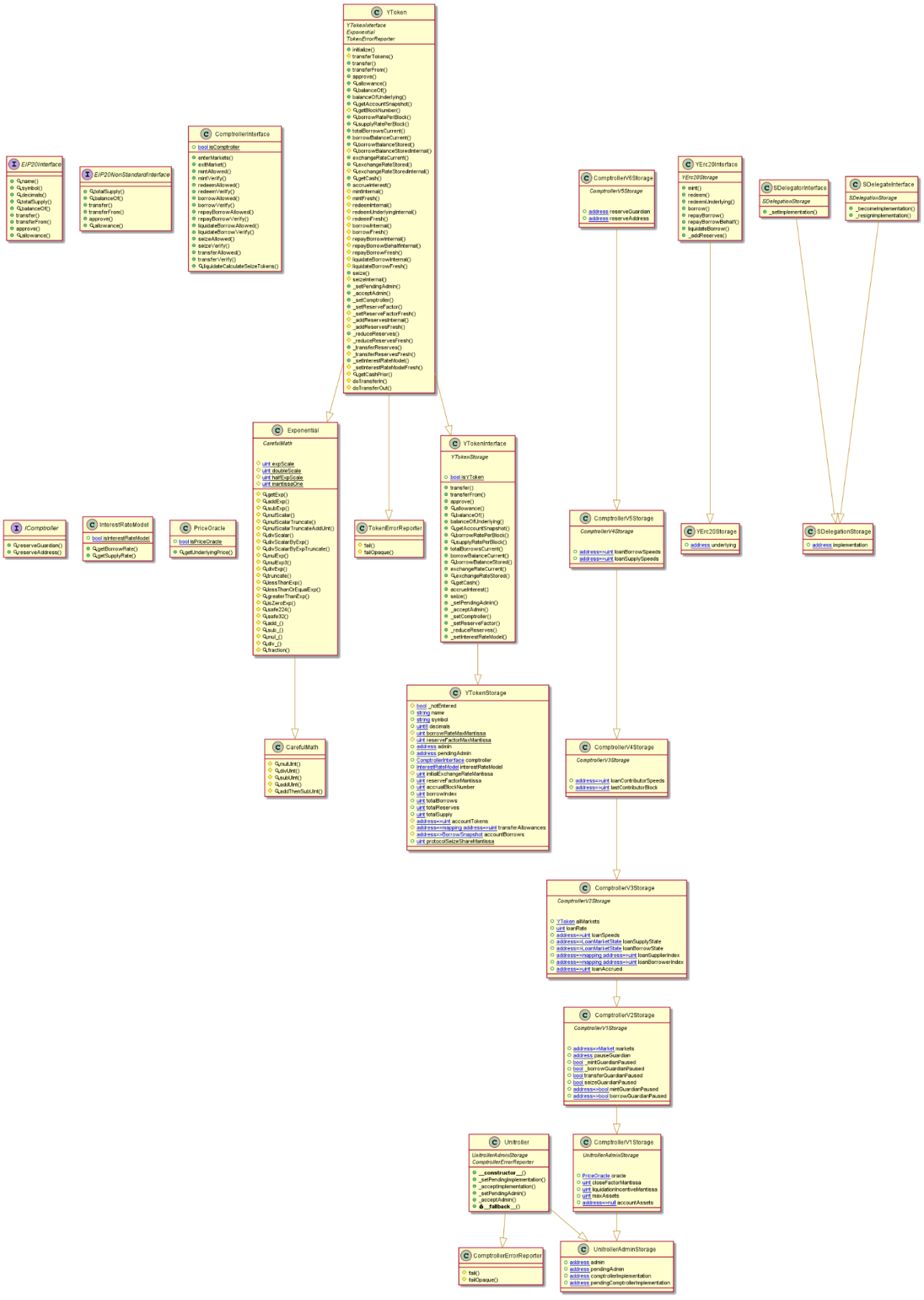
SimplePriceOracle Diagram



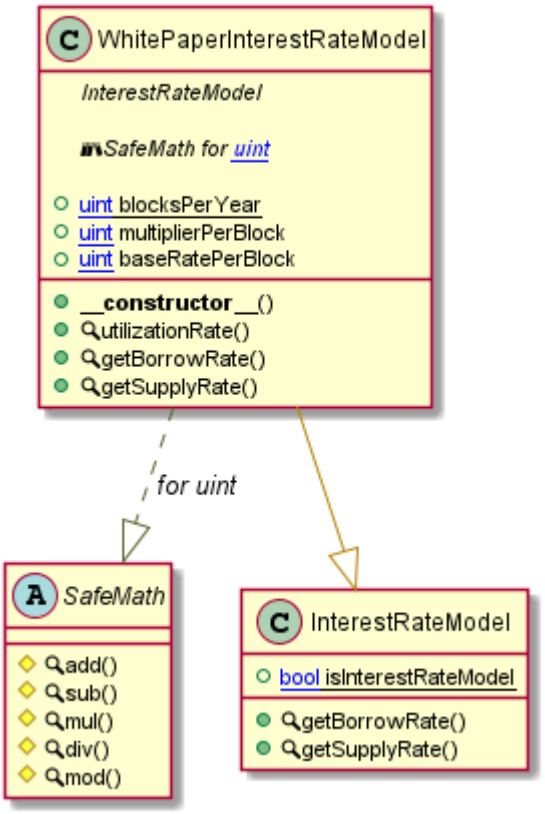
Timelock Diagram



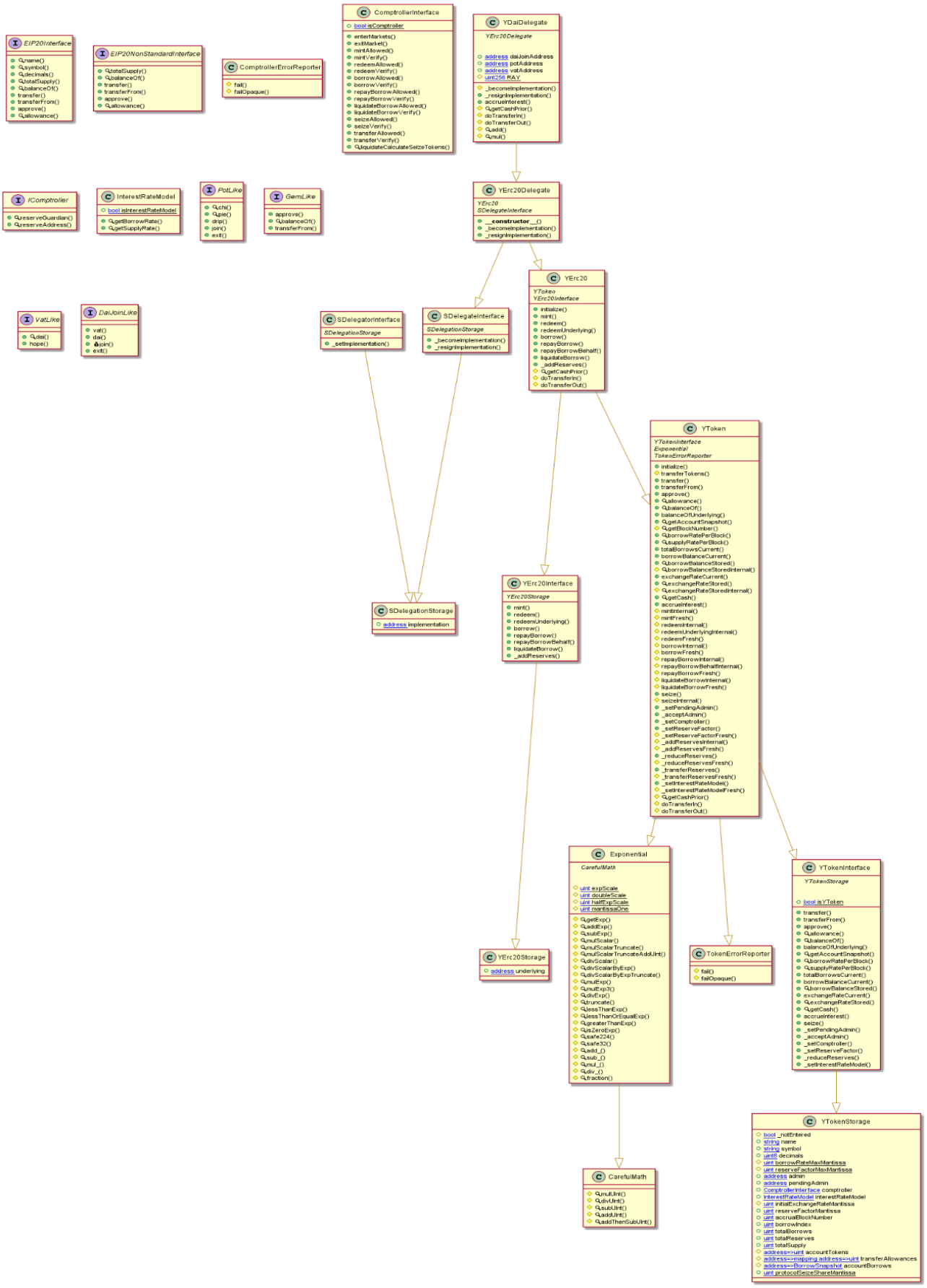
Unitroller Diagram



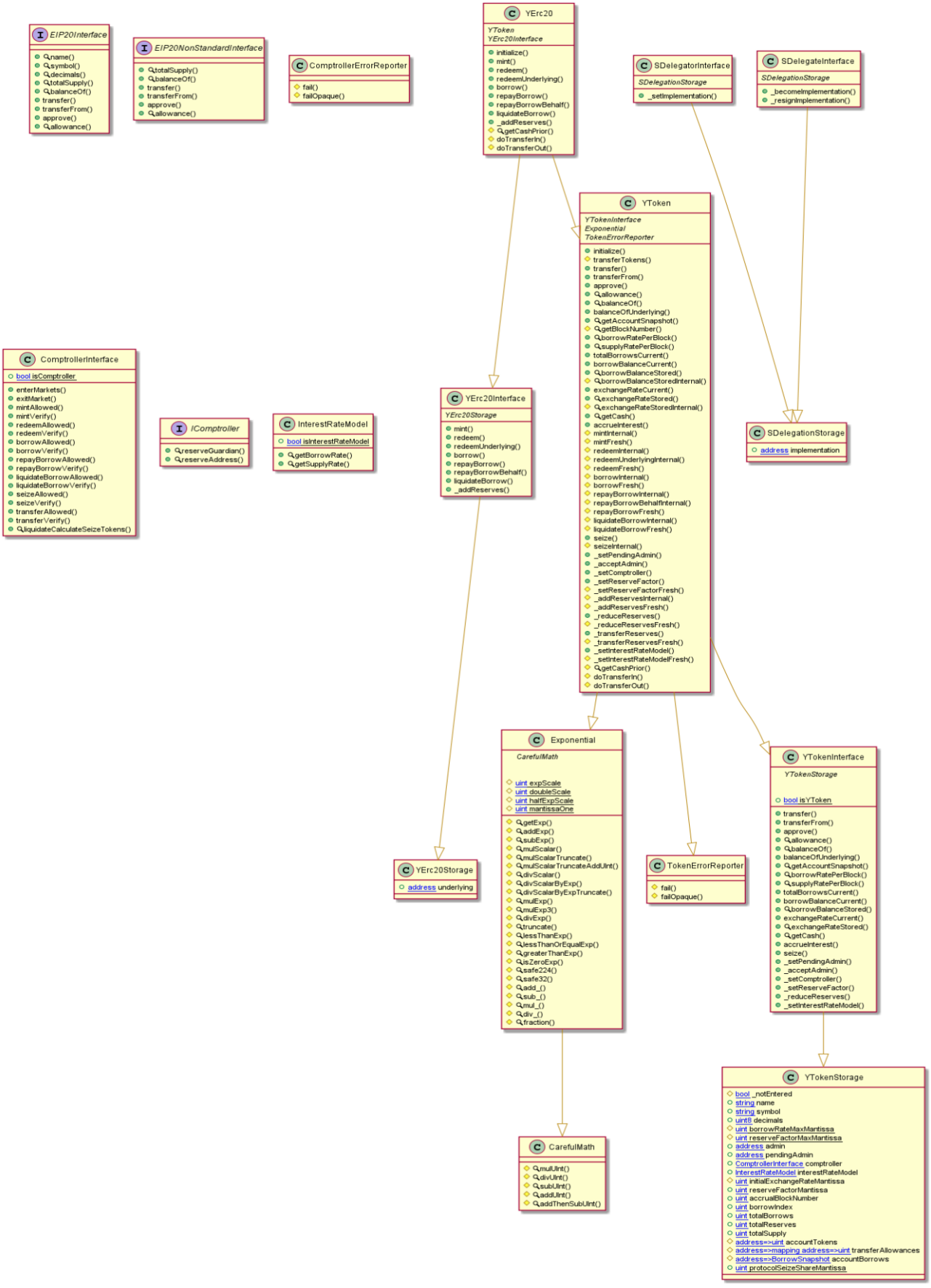
WhitePaperInterestRateModel Diagram



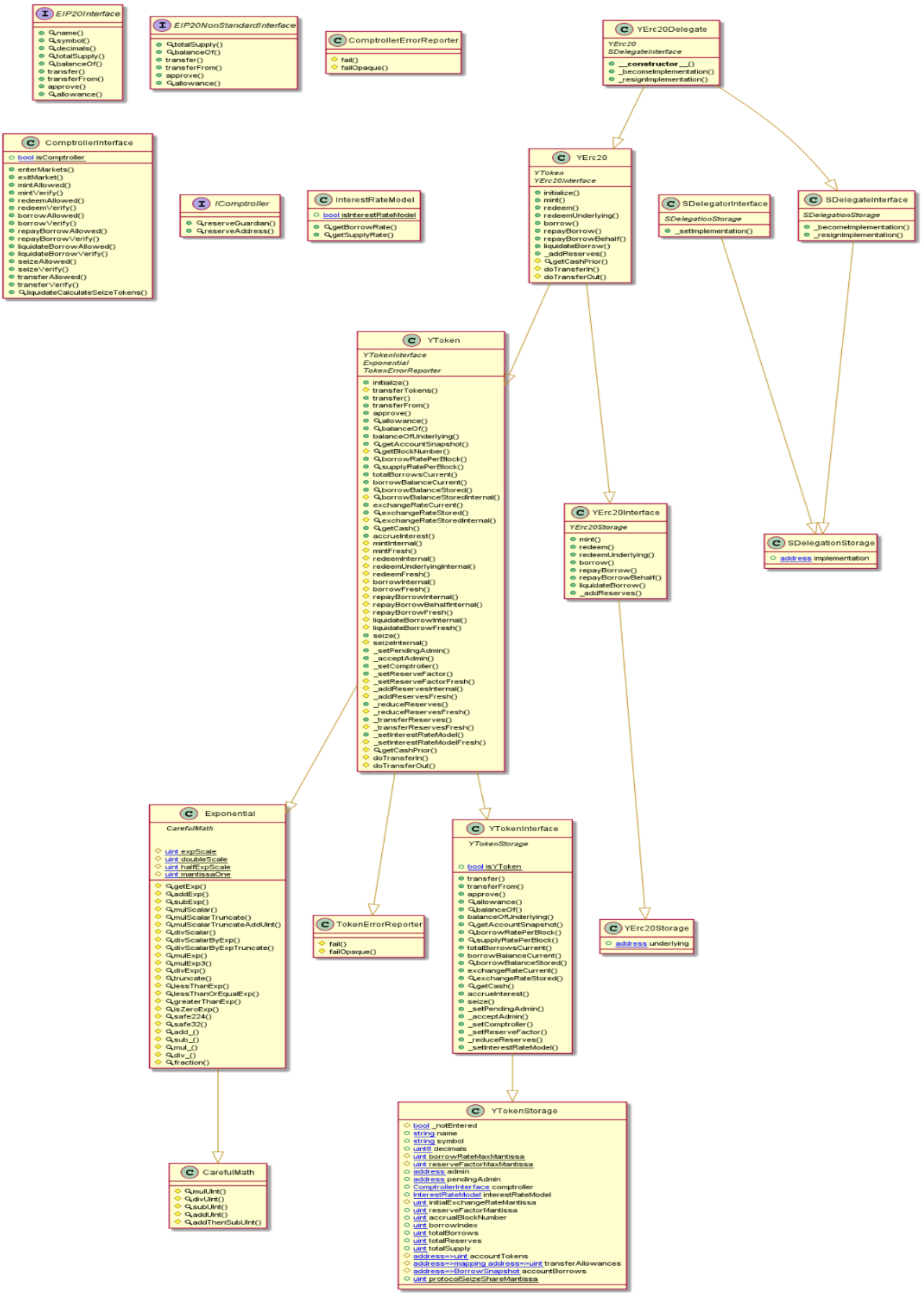
YDaiDelegate Diagram



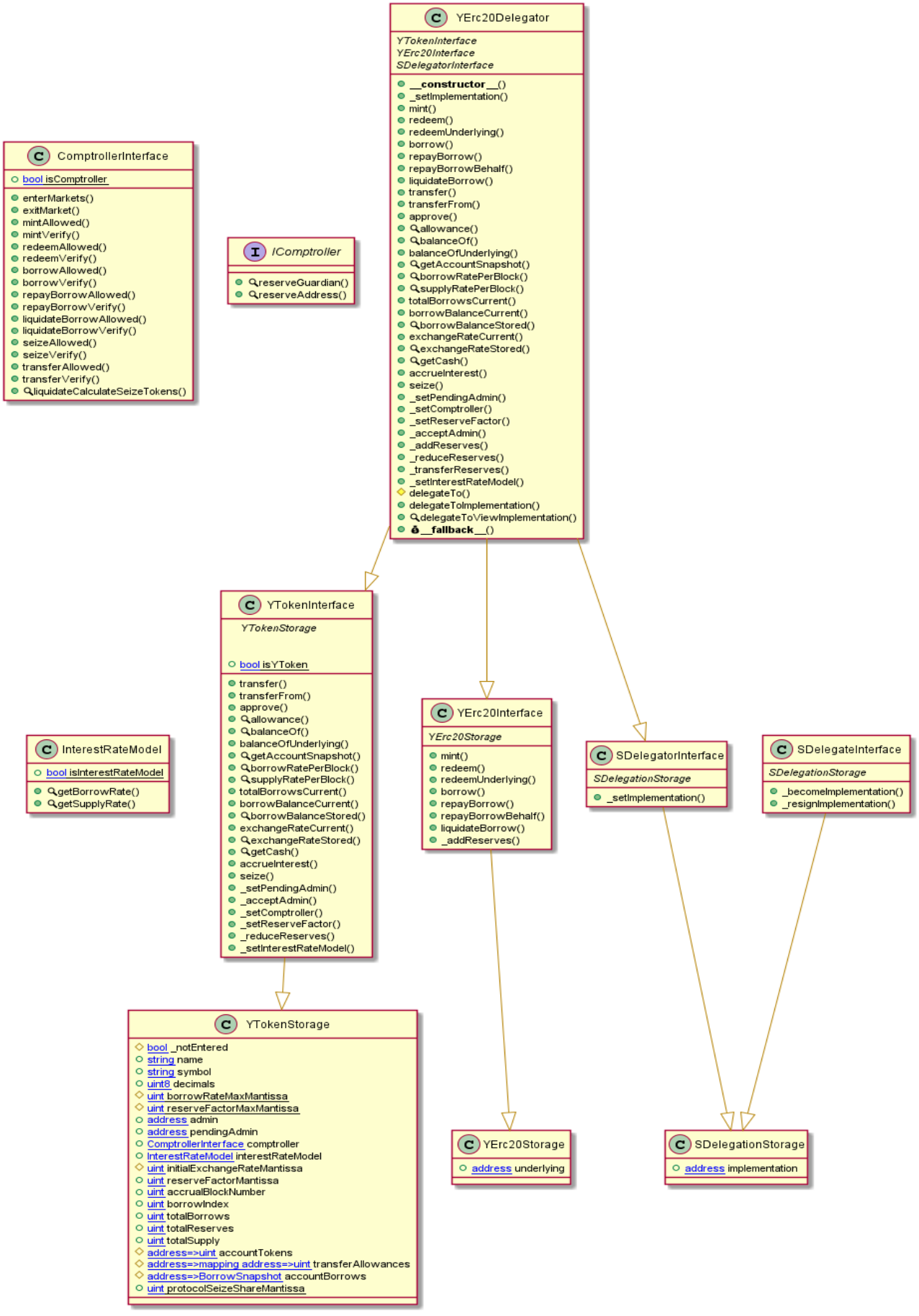
YErc20 Diagram



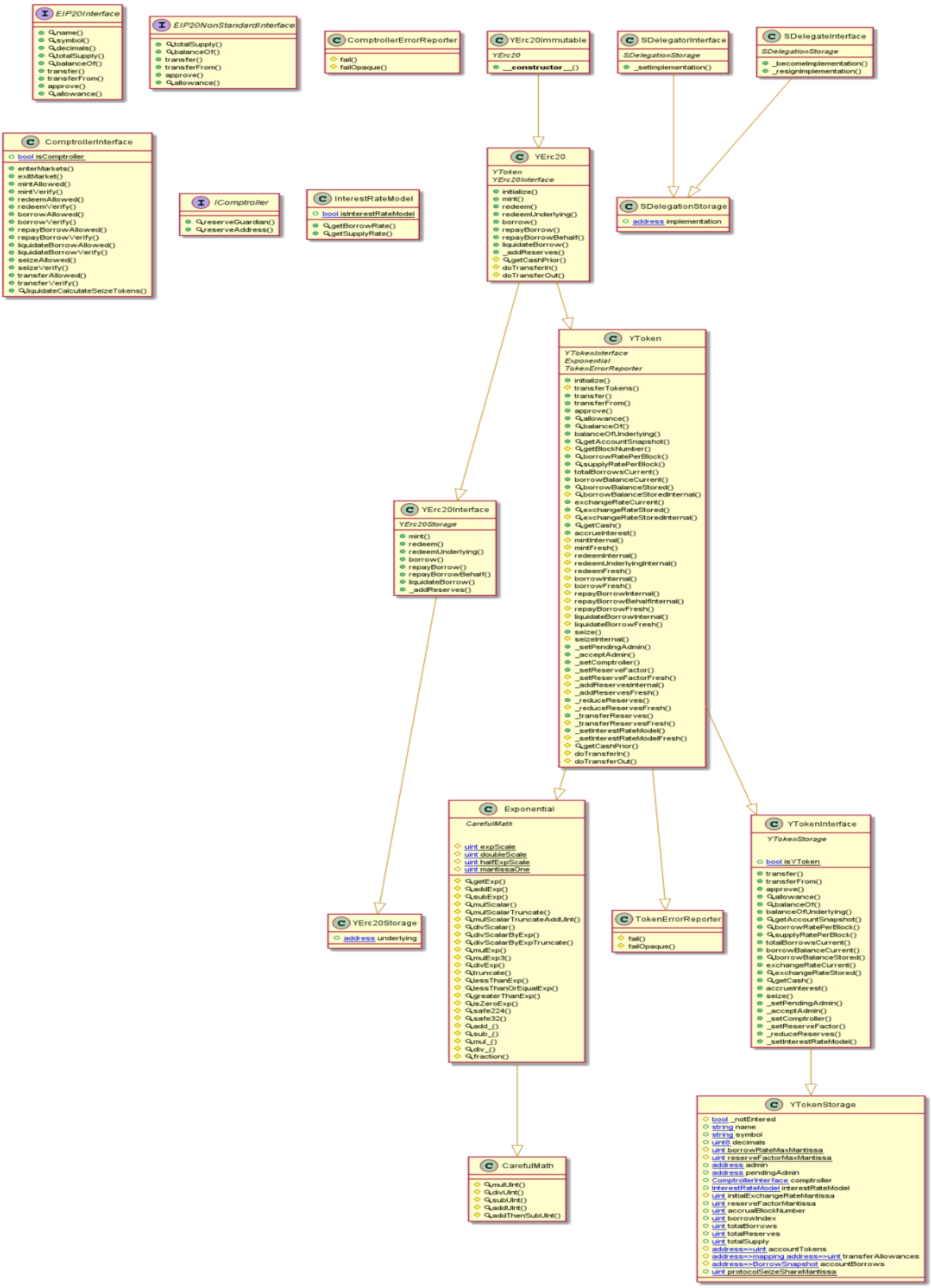
YErc20Delegate Diagram



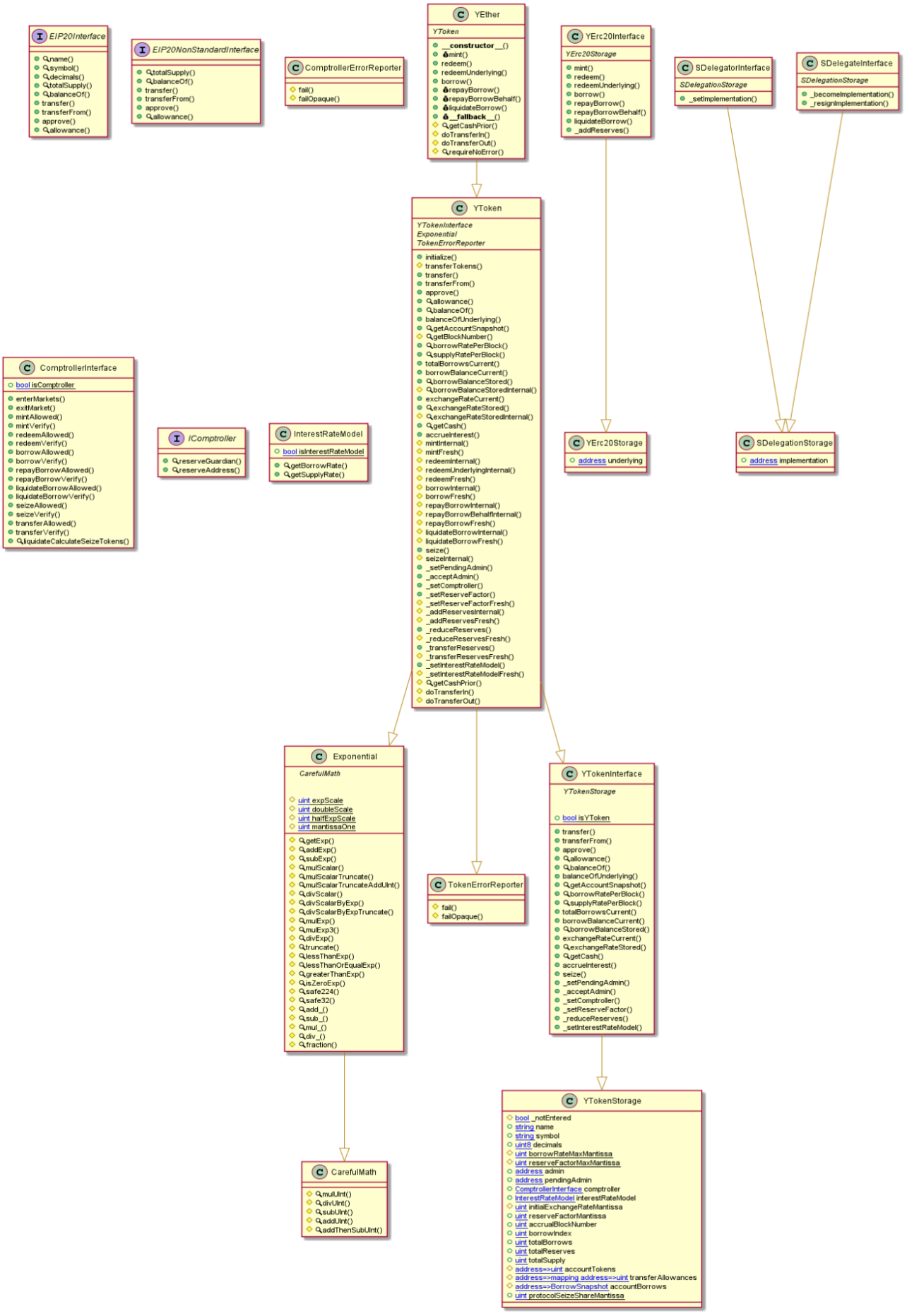
YErc20Delegator Diagram



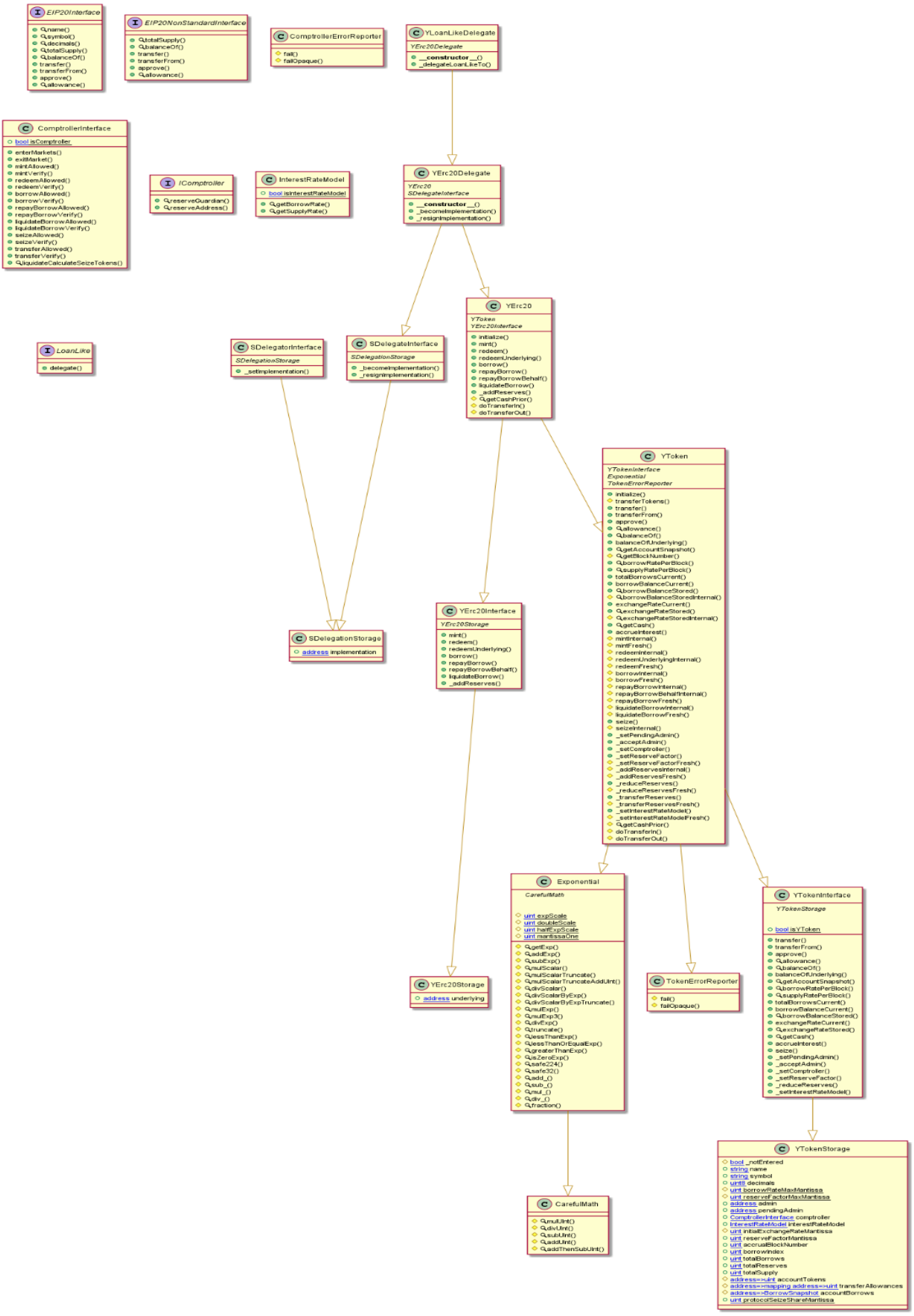
YErc20Immutable Diagram



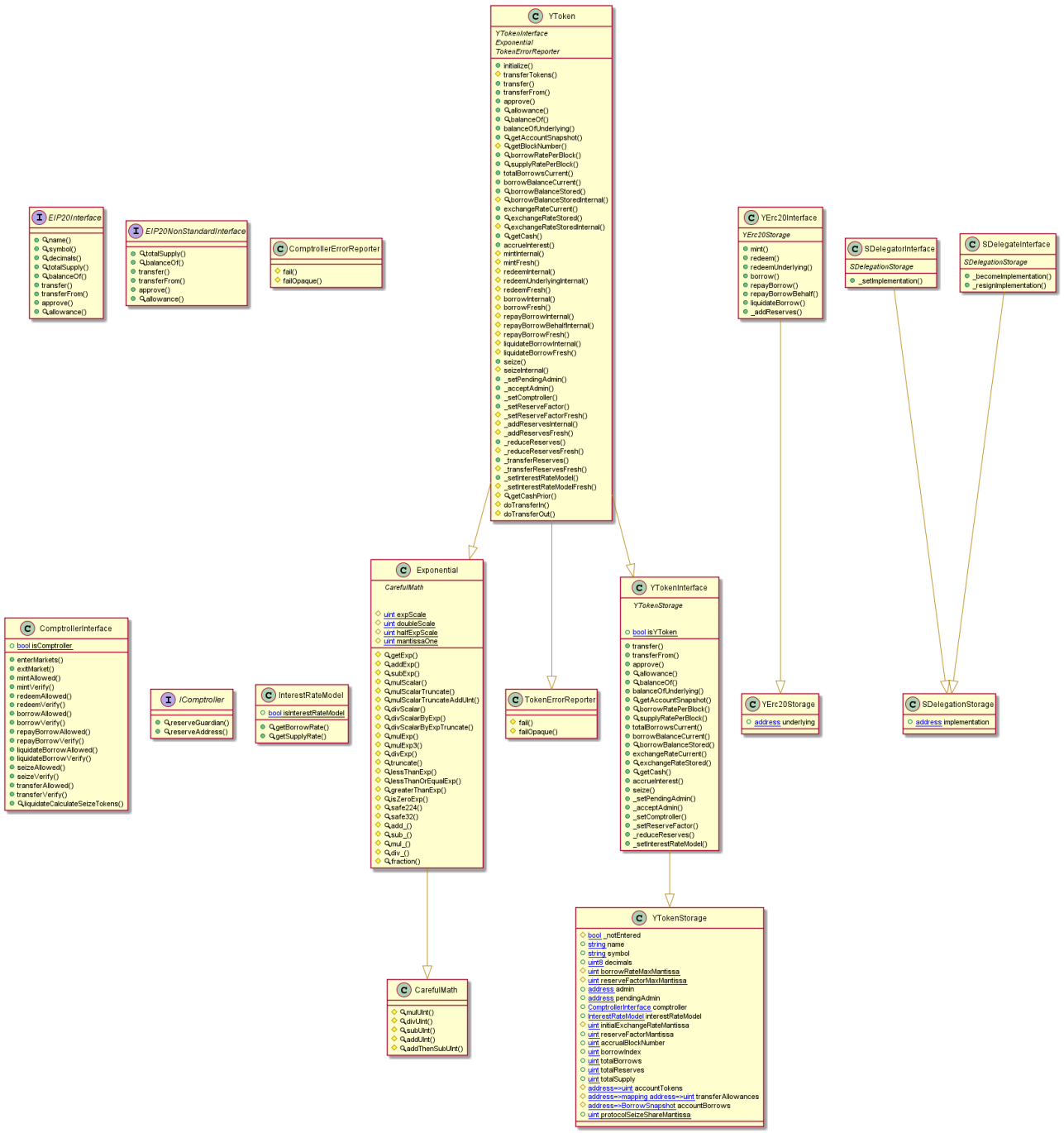
YEther Diagram



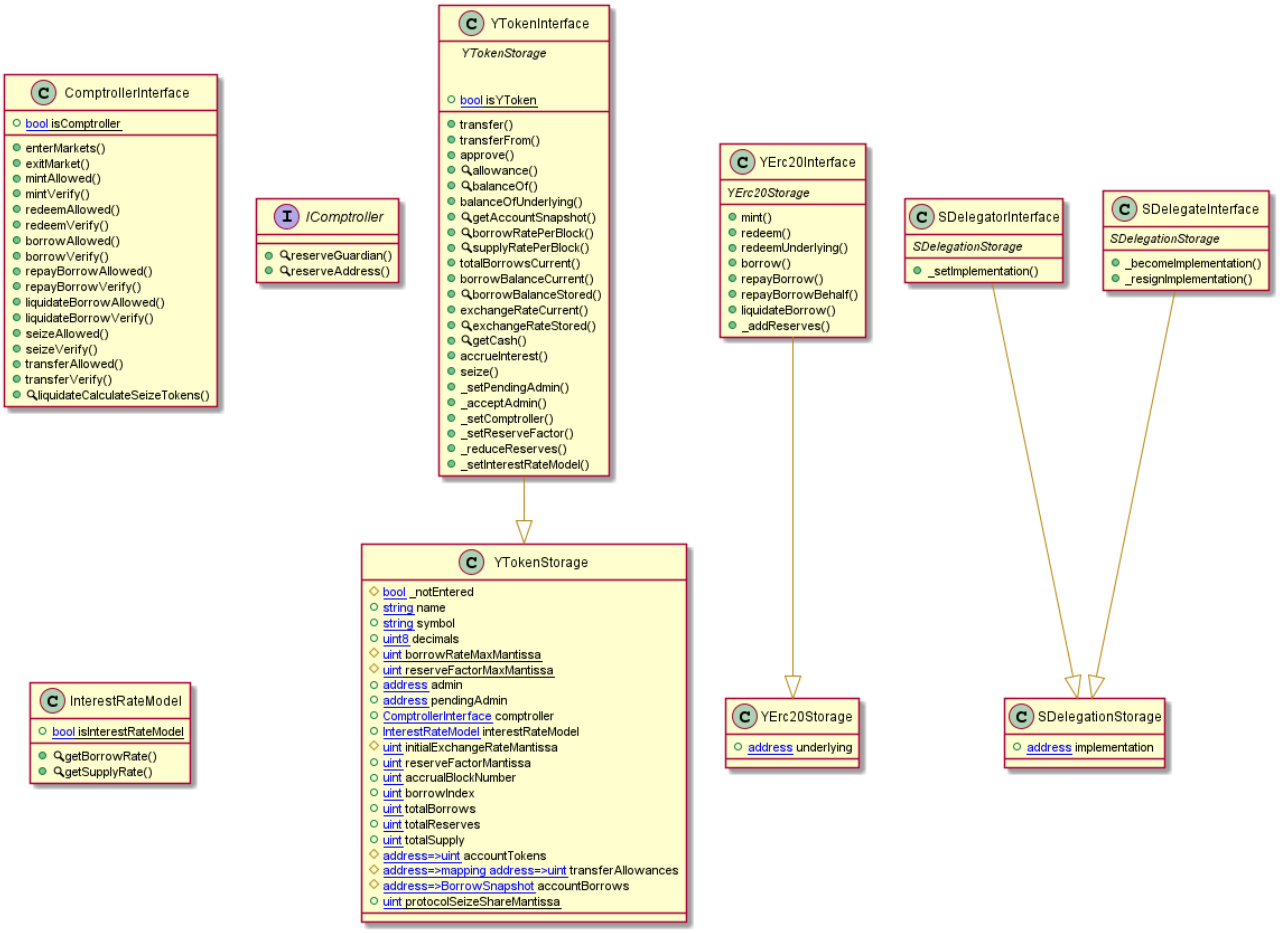
YLoanLikeDelegate Diagram



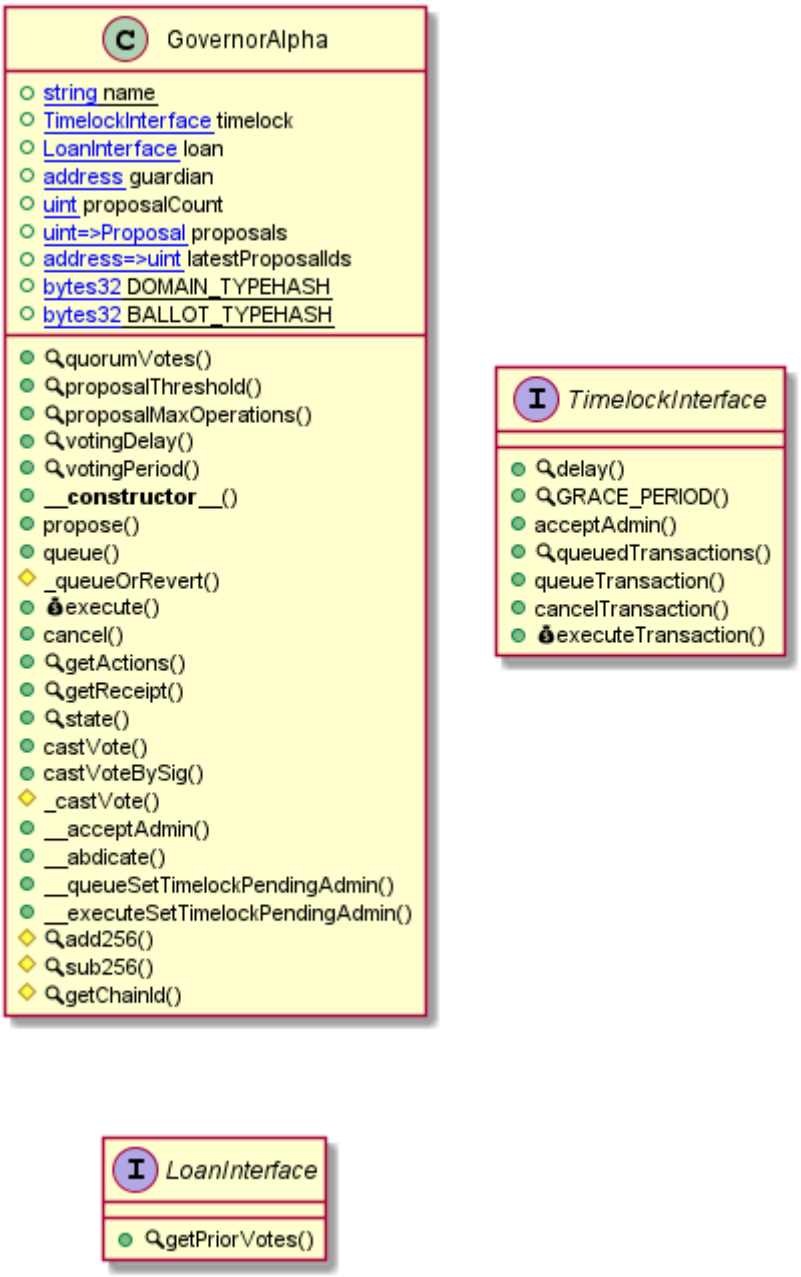
YToken Diagram



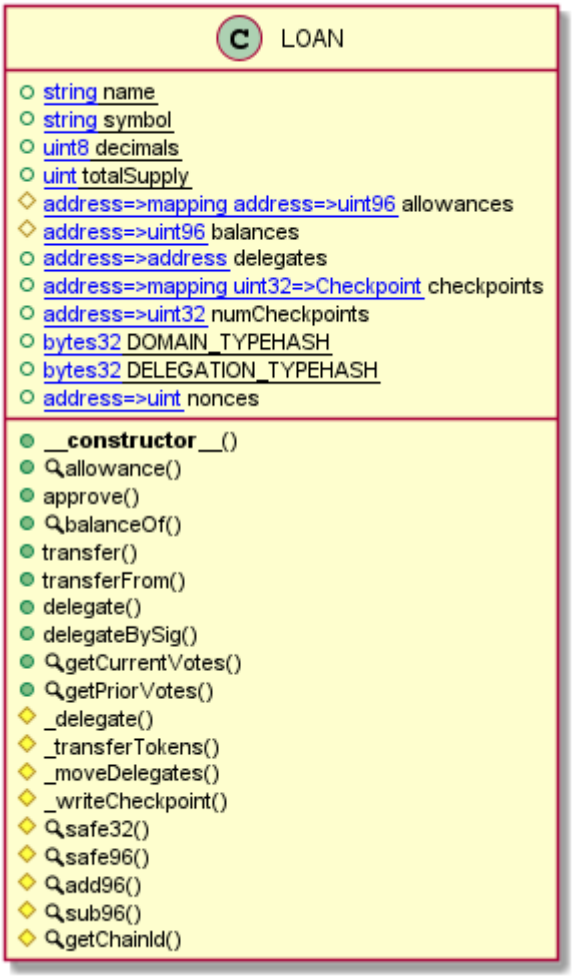
YTokenInterfaces Diagram



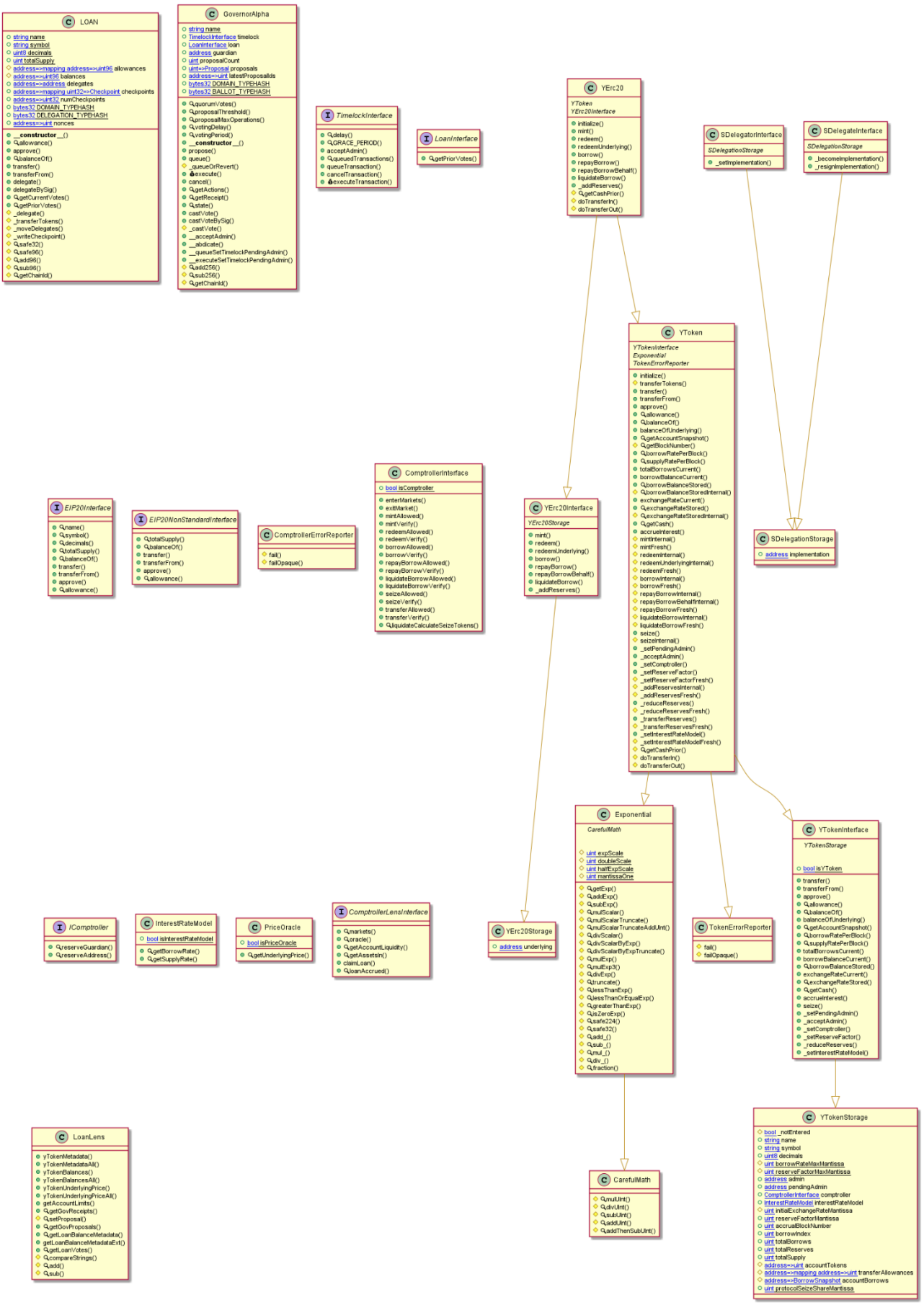
GovernorAlpha Diagram



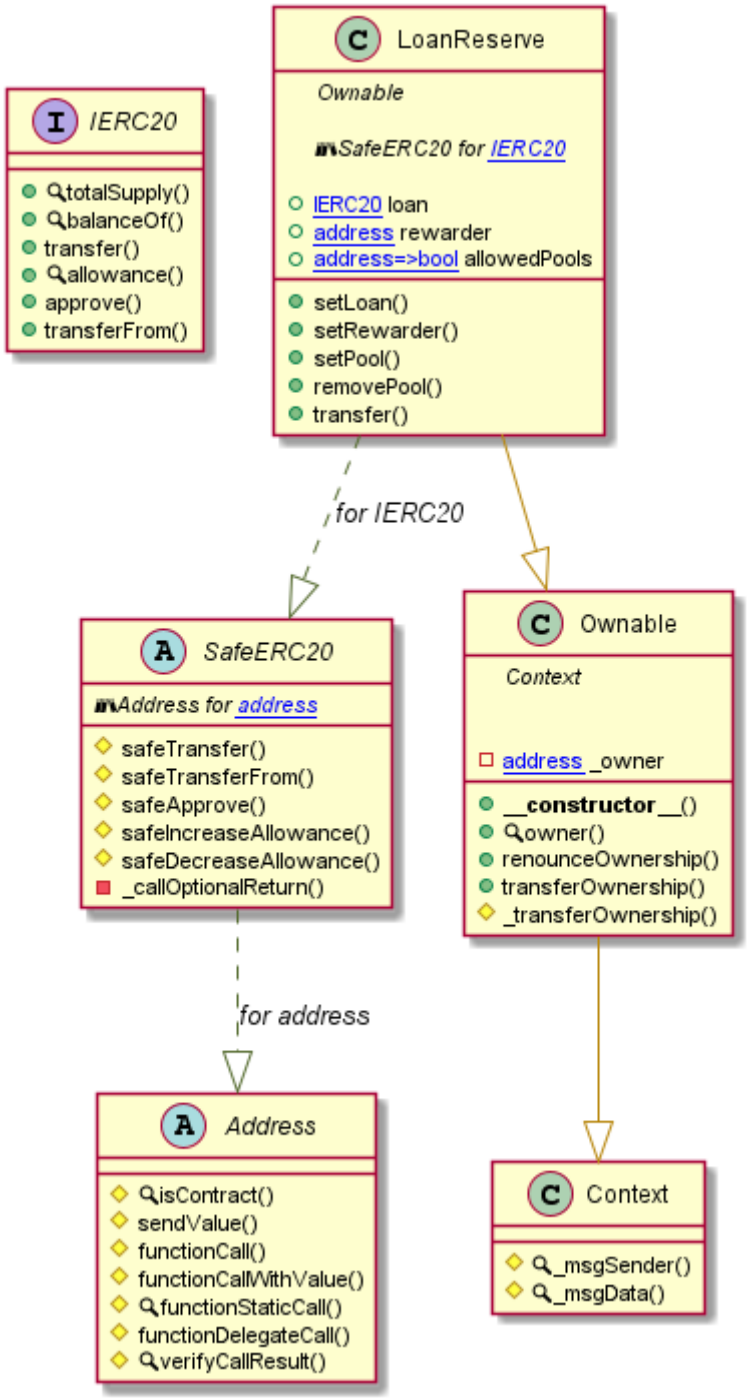
LOAN Diagram



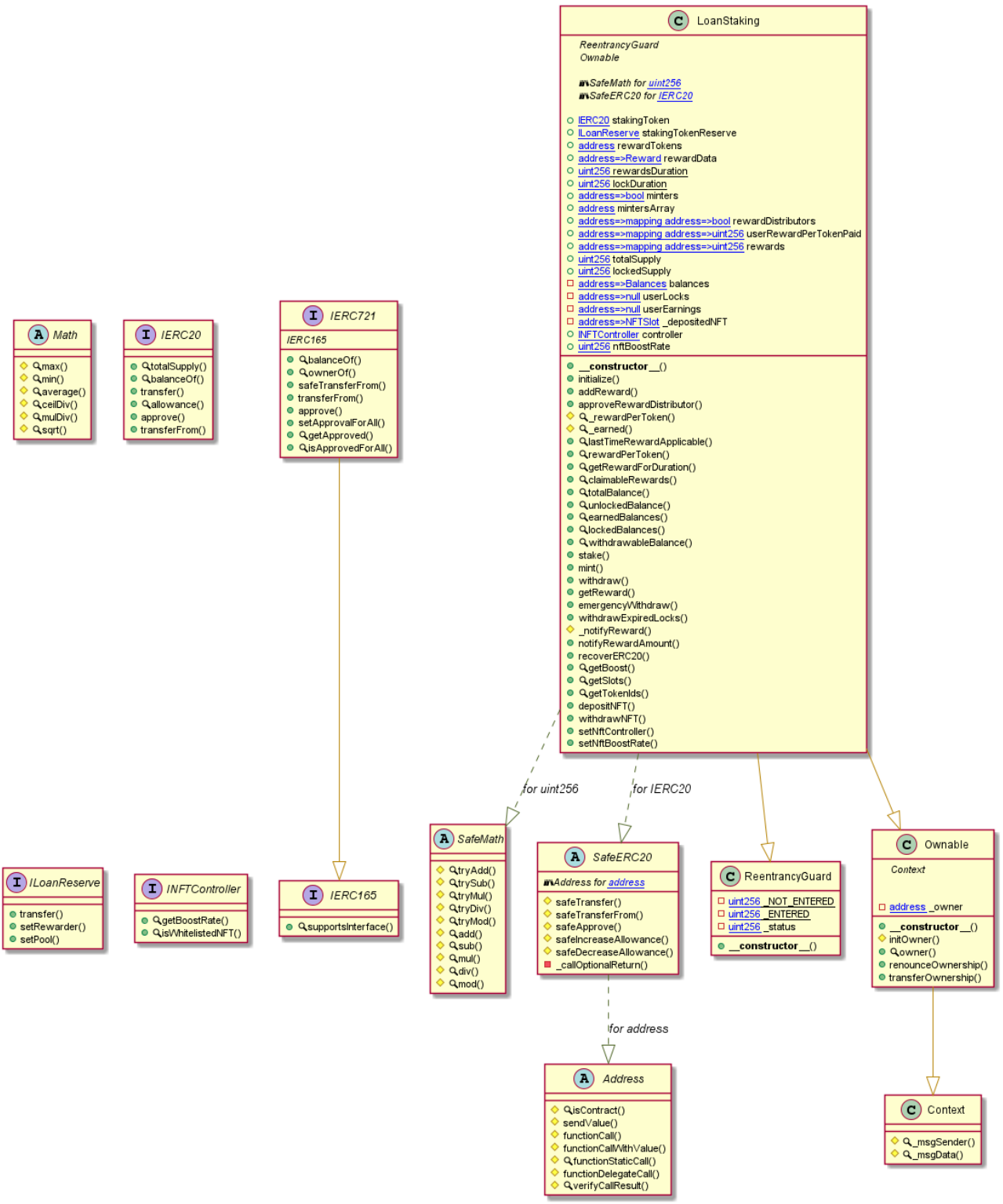
LoanLens Diagram



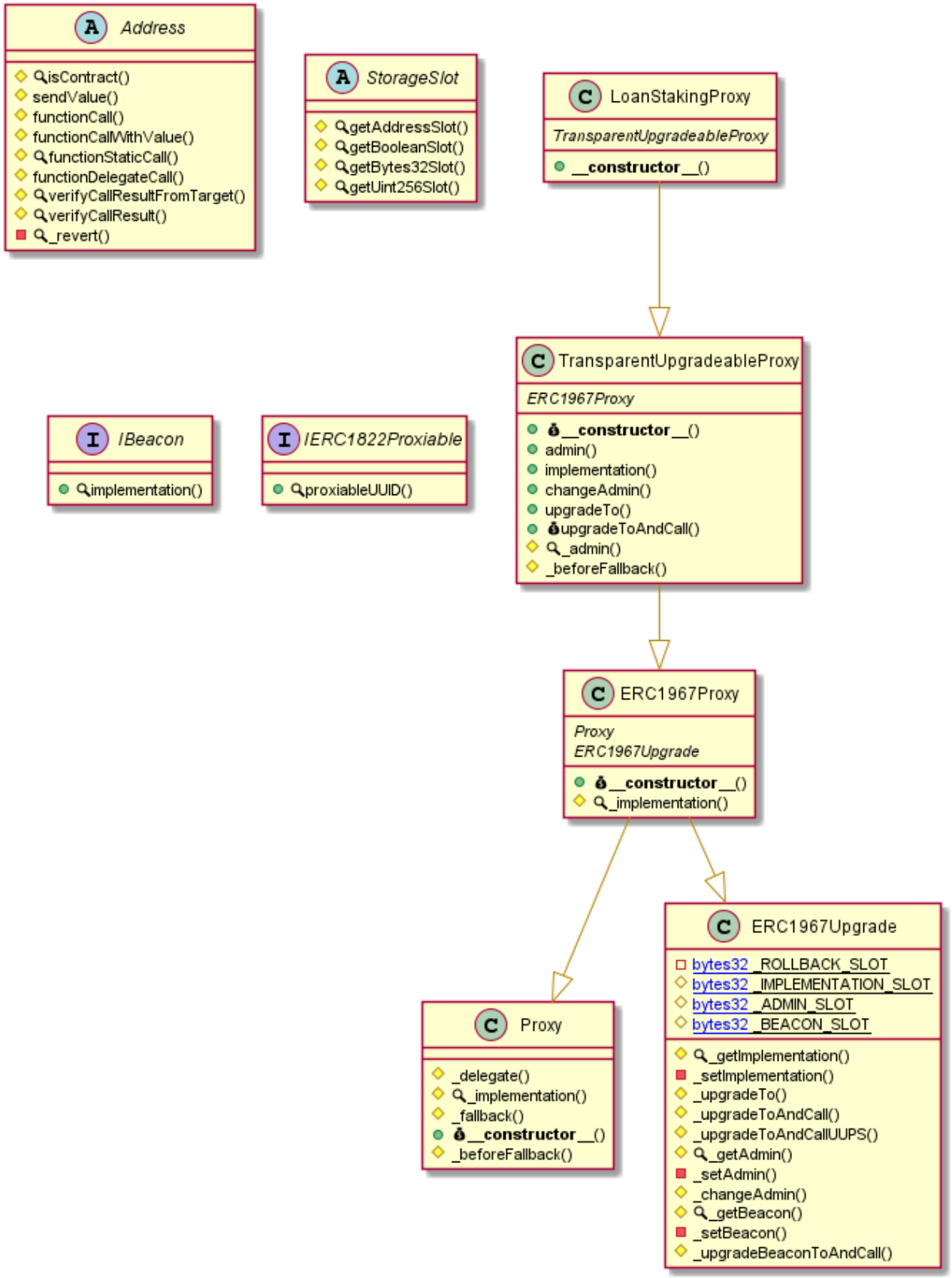
LoanReserve Diagram



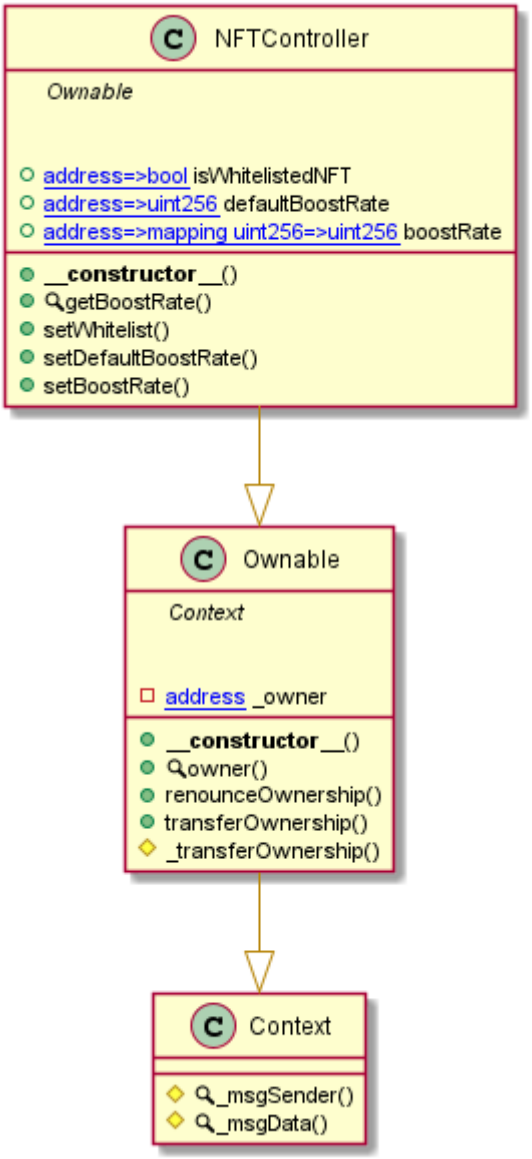
LoanStaking Diagram



LoanStakingProxy Diagram



NFTController Diagram



Slither Results Log

Slither log >> CarefulMath.sol

```
INFO:Detectors:
CarefulMath.addThenSubUInt(uint256,uint256,uint256) (CarefulMath.sol#71-79) is never used and should be removed
CarefulMath.addUInt(uint256,uint256) (CarefulMath.sol#58-66) is never used and should be removed
CarefulMath.divUInt(uint256,uint256) (CarefulMath.sol#36-42) is never used and should be removed
CarefulMath.mulUInt(uint256,uint256) (CarefulMath.sol#19-31) is never used and should be removed
CarefulMath.subUInt(uint256,uint256) (CarefulMath.sol#47-53) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Slither:CarefulMath.sol analyzed (1 contracts with 75 detectors), 5 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Comptroller.sol

```
INFO:Detectors:
delegate(address) should be declared external:
        - LOAN.delegate(address) (Comptroller.sol#163-165)
delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - LOAN.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (Comptroller.sol#167-176)
getPriorVotes(address,uint256) should be declared external:
        - LOAN.getPriorVotes(address,uint256) (Comptroller.sol#183-213)
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (Comptroller.sol#1914-1920)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (Comptroller.sol#981)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (Comptroller.sol#1011)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (Comptroller.sol#1015)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (Comptroller.sol#1017)
initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - YToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (Comptroller.sol#1021-1047)
_setPendingImplementation(address) should be declared external:
        - Unitroller._setPendingImplementation(address) (Comptroller.sol#2076-2089)
_acceptImplementation() should be declared external:
        - Unitroller._acceptImplementation() (Comptroller.sol#2091-2107)
_setPendingAdmin(address) should be declared external:
        - Unitroller._setPendingAdmin(address) (Comptroller.sol#2110-2122)
_acceptAdmin() should be declared external:
        - Unitroller._acceptAdmin() (Comptroller.sol#2124-2140)
enterMarkets(address[]) should be declared external:
        - Comptroller.enterMarkets(address[]) (Comptroller.sol#2228-2239)
getAccountLiquidity(address) should be declared external:
        - Comptroller.getAccountLiquidity(address) (Comptroller.sol#2593-2597)
```

```
        - Comptroller.getAccountLiquidity(address) (Comptroller.sol#2593-2597)
getHypotheticalAccountLiquidity(address,address,uint256,uint256) should be declared external:
        - Comptroller.getHypotheticalAccountLiquidity(address,address,uint256,uint256) (Comptroller.sol#2603-2610)
_setPriceOracle(PriceOracle) should be declared external:
        - Comptroller._setPriceOracle(PriceOracle) (Comptroller.sol#2712-2724)
_setPauseGuardian(address) should be declared external:
        - Comptroller._setPauseGuardian(address) (Comptroller.sol#2860-2872)
_setMintPaused(YToken,bool) should be declared external:
        - Comptroller._setMintPaused(YToken,bool) (Comptroller.sol#2874-2882)
_setBorrowPaused(YToken,bool) should be declared external:
        - Comptroller._setBorrowPaused(YToken,bool) (Comptroller.sol#2884-2892)
_setTransferPaused(bool) should be declared external:
        - Comptroller._setTransferPaused(bool) (Comptroller.sol#2894-2901)
_setSeizePaused(bool) should be declared external:
        - Comptroller._setSeizePaused(bool) (Comptroller.sol#2903-2910)
_become(Unitroller) should be declared external:
        - Comptroller._become(Unitroller) (Comptroller.sol#2928-2931)
_setLoanSpeeds(YToken[],uint256[],uint256[]) should be declared external:
        - Comptroller._setLoanSpeeds(YToken[],uint256[],uint256[]) (Comptroller.sol#2938-2947)
claimLoan(address) should be declared external:
        - Comptroller.claimLoan(address) (Comptroller.sol#3053-3055)
_setContributorLoanSpeed(address,uint256) should be declared external:
        - Comptroller._setContributorLoanSpeed(address,uint256) (Comptroller.sol#3112-3125)
_grantLOAN(address,uint256) should be declared external:
        - Comptroller._grantLOAN(address,uint256) (Comptroller.sol#3127-3134)
_setLoanRate(uint256) should be declared external:
        - Comptroller._setLoanRate(uint256) (Comptroller.sol#3136-3143)
_dropLoanMarket(address) should be declared external:
        - Comptroller._dropLoanMarket(address) (Comptroller.sol#3145-3154)
getAllMarkets() should be declared external:
        - Comptroller.getAllMarkets() (Comptroller.sol#3156-3158)
canClaimLoanBySuppling(address) should be declared external:
        - Comptroller.canClaimLoanBySuppling(address) (Comptroller.sol#3169-3178)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Comptroller.sol analyzed (28 contracts with 75 detectors), 276 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> ComptrollerG1.sol

```
INFO:Detectors:
delegate(address) should be declared external:
        - LOAN.delegate(address) (ComptrollerG1.sol#163-165)
delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - LOAN.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (ComptrollerG1.sol#167-176)
getPriorVotes(address,uint256) should be declared external:
        - LOAN.getPriorVotes(address,uint256) (ComptrollerG1.sol#183-213)
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (ComptrollerG1.sol#1914-1920)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (ComptrollerG1.sol#981)
```

```
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (ComptrollerG1.sol#1914-1920)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (ComptrollerG1.sol#981)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (ComptrollerG1.sol#1011)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (ComptrollerG1.sol#1015)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (ComptrollerG1.sol#1017)
initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - YToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (ComptrollerG1.sol#1021-1047)
_setPendingImplementation(address) should be declared external:
        - Unitroller._setPendingImplementation(address) (ComptrollerG1.sol#2076-2089)
_acceptImplementation() should be declared external:
        - Unitroller._acceptImplementation() (ComptrollerG1.sol#2091-2107)
_setPendingAdmin(address) should be declared external:
        - Unitroller._setPendingAdmin(address) (ComptrollerG1.sol#2110-2122)
_acceptAdmin() should be declared external:
        - Unitroller._acceptAdmin() (ComptrollerG1.sol#2124-2140)
enterMarkets(address[]) should be declared external:
        - ComptrollerG1.enterMarkets(address[]) (ComptrollerG1.sol#2209-2241)
getAccountLiquidity(address) should be declared external:
        - ComptrollerG1.getAccountLiquidity(address) (ComptrollerG1.sol#2543-2547)
_setPriceOracle(PriceOracle) should be declared external:
        - ComptrollerG1._setPriceOracle(PriceOracle) (ComptrollerG1.sol#2653-2666)
_become(Unitroller,PriceOracle,uint256,uint256,bool) should be declared external:
        - ComptrollerG1._become(Unitroller,PriceOracle,uint256,uint256,bool) (ComptrollerG1.sol#2774-2795)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ComptrollerG1.sol analyzed (28 contracts with 75 detectors), 255 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> ComptrollerG2.sol

```
INFO:Detectors:
delegate(address) should be declared external:
        - LOAN.delegate(address) (ComptrollerG2.sol#163-165)
delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - LOAN.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (ComptrollerG2.sol#167-176)
getPriorVotes(address,uint256) should be declared external:
        - LOAN.getPriorVotes(address,uint256) (ComptrollerG2.sol#183-213)
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (ComptrollerG2.sol#1914-1920)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (ComptrollerG2.sol#981)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (ComptrollerG2.sol#1011)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (ComptrollerG2.sol#1015)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (ComptrollerG2.sol#1017)
initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - YToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (ComptrollerG2.sol#1021-1047)
_setPendingImplementation(address) should be declared external:
        - Unitroller._setPendingImplementation(address) (ComptrollerG2.sol#2076-2089)
_acceptImplementation() should be declared external:
        - Unitroller._acceptImplementation() (ComptrollerG2.sol#2091-2107)
_setPendingAdmin(address) should be declared external:
        - Unitroller._setPendingAdmin(address) (ComptrollerG2.sol#2110-2122)
_acceptAdmin() should be declared external:
        - Unitroller._acceptAdmin() (ComptrollerG2.sol#2124-2140)
enterMarkets(address[]) should be declared external:
        - ComptrollerG2.enterMarkets(address[]) (ComptrollerG2.sol#2204-2215)
getAccountLiquidity(address) should be declared external:
        - ComptrollerG2.getAccountLiquidity(address) (ComptrollerG2.sol#2548-2552)
getHypotheticalAccountLiquidity(address,address,uint256,uint256) should be declared external:
        - ComptrollerG2.getHypotheticalAccountLiquidity(address,address,uint256,uint256) (ComptrollerG2.sol#2558-2565)
_setPriceOracle(PriceOracle) should be declared external:
        - ComptrollerG2._setPriceOracle(PriceOracle) (ComptrollerG2.sol#2667-2679)
```

```
_setPriceOracle(PriceOracle) should be declared external:
        - ComptrollerG2._setPriceOracle(PriceOracle) (ComptrollerG2.sol#2667-2679)
_setPauseGuardian(address) should be declared external:
        - ComptrollerG2._setPauseGuardian(address) (ComptrollerG2.sol#2787-2799)
_setMintPaused(YToken,bool) should be declared external:
        - ComptrollerG2._setMintPaused(YToken,bool) (ComptrollerG2.sol#2801-2809)
_setBorrowPaused(YToken,bool) should be declared external:
        - ComptrollerG2._setBorrowPaused(YToken,bool) (ComptrollerG2.sol#2811-2819)
_setTransferPaused(bool) should be declared external:
        - ComptrollerG2._setTransferPaused(bool) (ComptrollerG2.sol#2821-2828)
_setSeizePaused(bool) should be declared external:
        - ComptrollerG2._setSeizePaused(bool) (ComptrollerG2.sol#2830-2837)
_become(Unitroller) should be declared external:
        - ComptrollerG2._become(Unitroller) (ComptrollerG2.sol#2839-2844)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ComptrollerG2.sol analyzed (28 contracts with 75 detectors), 260 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> ComptrollerG3.sol

```
INFO:Detectors:
delegate(address) should be declared external:
        - LOAN.delegate(address) (ComptrollerG3.sol#163-165)
delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - LOAN.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (ComptrollerG3.sol#167-176)
getPriorVotes(address,uint256) should be declared external:
        - LOAN.getPriorVotes(address,uint256) (ComptrollerG3.sol#183-213)
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (ComptrollerG3.sol#1914-1920)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (ComptrollerG3.sol#981)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (ComptrollerG3.sol#1011)
```

```
        - YTokenInterface._setInterestRateModel(InterestRateModel) (ComptrollerG3.sol#981)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (ComptrollerG3.sol#1011)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (ComptrollerG3.sol#1015)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (ComptrollerG3.sol#1017)
initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - YToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (ComptrollerG3.sol#1021-1047)
_setPendingImplementation(address) should be declared external:
        - Unitroller._setPendingImplementation(address) (ComptrollerG3.sol#2076-2089)
_acceptImplementation() should be declared external:
        - Unitroller._acceptImplementation() (ComptrollerG3.sol#2091-2107)
_setPendingAdmin(address) should be declared external:
        - Unitroller._setPendingAdmin(address) (ComptrollerG3.sol#2110-2122)
_acceptAdmin() should be declared external:
        - Unitroller._acceptAdmin() (ComptrollerG3.sol#2124-2140)
enterMarkets(address[]) should be declared external:
        - ComptrollerG3.enterMarkets(address[]) (ComptrollerG3.sol#2219-2230)
getAccountLiquidity(address) should be declared external:
        - ComptrollerG3.getAccountLiquidity(address) (ComptrollerG3.sol#2584-2588)
getHypotheticalAccountLiquidity(address,address,uint256,uint256) should be declared external:
        - ComptrollerG3.getHypotheticalAccountLiquidity(address,address,uint256,uint256) (ComptrollerG3.sol#2594-2601)
_setPriceOracle(PriceOracle) should be declared external:
        - ComptrollerG3._setPriceOracle(PriceOracle) (ComptrollerG3.sol#2703-2715)
_setPauseGuardian(address) should be declared external:
        - ComptrollerG3._setPauseGuardian(address) (ComptrollerG3.sol#2833-2845)
_setMintPaused(YToken,bool) should be declared external:
        - ComptrollerG3._setMintPaused(YToken,bool) (ComptrollerG3.sol#2847-2855)
_setBorrowPaused(YToken,bool) should be declared external:
        - ComptrollerG3._setBorrowPaused(YToken,bool) (ComptrollerG3.sol#2857-2865)
_setTransferPaused(bool) should be declared external:
        - ComptrollerG3._setTransferPaused(bool) (ComptrollerG3.sol#2867-2874)
_setSeizePaused(bool) should be declared external:
        - ComptrollerG3._setSeizePaused(bool) (ComptrollerG3.sol#2876-2883)
_become(Unitroller,uint256,address[],address[]) should be declared external:
        - ComptrollerG3._become(Unitroller,uint256,address[],address[]) (ComptrollerG3.sol#2885-2890)
```

```
_become(Unitroller,uint256,address[],address[]) should be declared external:
        - ComptrollerG3._become(Unitroller,uint256,address[],address[]) (ComptrollerG3.sol#2885-2890)
_becomeG3(uint256,address[],address[]) should be declared external:
        - ComptrollerG3._becomeG3(uint256,address[],address[]) (ComptrollerG3.sol#2892-2905)
claimLoan(address) should be declared external:
        - ComptrollerG3.claimLoan(address) (ComptrollerG3.sol#3027-3029)
_dropLoanMarket(address) should be declared external:
        - ComptrollerG3._dropLoanMarket(address) (ComptrollerG3.sol#3101-3111)
getAllMarkets() should be declared external:
        - ComptrollerG3.getAllMarkets() (ComptrollerG3.sol#3113-3115)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ComptrollerG3.sol analyzed (28 contracts with 75 detectors), 275 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> ComptrollerG4.sol

```
INFO:Detectors:
delegate(address) should be declared external:
        - LOAN.delegate(address) (ComptrollerG4.sol#163-165)
delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - LOAN.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (ComptrollerG4.sol#167-176)
getPriorVotes(address,uint256) should be declared external:
        - LOAN.getPriorVotes(address,uint256) (ComptrollerG4.sol#183-213)
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (ComptrollerG4.sol#1914-1920)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (ComptrollerG4.sol#981)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (ComptrollerG4.sol#1011)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (ComptrollerG4.sol#1015)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (ComptrollerG4.sol#1017)
initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - YToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (ComptrollerG4.sol#1021-1047)
_setPendingImplementation(address) should be declared external:
        - Unitroller._setPendingImplementation(address) (ComptrollerG4.sol#2076-2089)
_acceptImplementation() should be declared external:
        - Unitroller._acceptImplementation() (ComptrollerG4.sol#2091-2107)
_setPendingAdmin(address) should be declared external:
        - Unitroller._setPendingAdmin(address) (ComptrollerG4.sol#2110-2122)
_acceptAdmin() should be declared external:
        - Unitroller._acceptAdmin() (ComptrollerG4.sol#2124-2140)
enterMarkets(address[]) should be declared external:
        - ComptrollerG4.enterMarkets(address[]) (ComptrollerG4.sol#2218-2229)
getAccountLiquidity(address) should be declared external:
        - ComptrollerG4.getAccountLiquidity(address) (ComptrollerG4.sol#2583-2587)
getHypotheticalAccountLiquidity(address,address,uint256,uint256) should be declared external:
        - ComptrollerG4.getHypotheticalAccountLiquidity(address,address,uint256,uint256) (ComptrollerG4.sol#2593-2600)
_setPriceOracle(PriceOracle) should be declared external:
```

```
_setLoanSpeed(YToken,uint256) should be declared external:
        - ComptrollerG4._setLoanSpeed(YToken,uint256) (ComptrollerG4.sol#2894-2897)
claimLoan(address) should be declared external:
        - ComptrollerG4.claimLoan(address) (ComptrollerG4.sol#3017-3019)
_setLoanRate(uint256) should be declared external:
        - ComptrollerG4._setLoanRate(uint256) (ComptrollerG4.sol#3060-3067)
_addLoanMarkets(address[]) should be declared external:
        - ComptrollerG4._addLoanMarkets(address[]) (ComptrollerG4.sol#3069-3076)
_dropLoanMarket(address) should be declared external:
        - ComptrollerG4._dropLoanMarket(address) (ComptrollerG4.sol#3101-3110)
getAllMarkets() should be declared external:
        - ComptrollerG4.getAllMarkets() (ComptrollerG4.sol#3112-3114)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ComptrollerG4.sol analyzed (28 contracts with 75 detectors), 273 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> ComptrollerG5.sol

```
INFO:Detectors:
delegate(address) should be declared external:
        - LOAN.delegate(address) (ComptrollerG5.sol#163-165)
delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - LOAN.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (ComptrollerG5.sol#167-176)
getPriorVotes(address,uint256) should be declared external:
        - LOAN.getPriorVotes(address,uint256) (ComptrollerG5.sol#183-213)
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (ComptrollerG5.sol#1914-1920)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (ComptrollerG5.sol#981)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (ComptrollerG5.sol#1011)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (ComptrollerG5.sol#1015)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (ComptrollerG5.sol#1017)
initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - YToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (ComptrollerG5.sol#1021-1047)
_setPendingImplementation(address) should be declared external:
        - Unitroller._setPendingImplementation(address) (ComptrollerG5.sol#2076-2089)
_acceptImplementation() should be declared external:
        - Unitroller._acceptImplementation() (ComptrollerG5.sol#2091-2107)
_setPendingAdmin(address) should be declared external:
        - Unitroller._setPendingAdmin(address) (ComptrollerG5.sol#2110-2122)
_acceptAdmin() should be declared external:
        - Unitroller._acceptAdmin() (ComptrollerG5.sol#2124-2140)
enterMarkets(address[]) should be declared external:
        - Comptroller.enterMarkets(address[]) (ComptrollerG5.sol#2226-2237)
getAccountLiquidity(address) should be declared external:
        - Comptroller.getAccountLiquidity(address) (ComptrollerG5.sol#2591-2595)
getHypotheticalAccountLiquidity(address,address,uint256,uint256) should be declared external:
        - Comptroller.getHypotheticalAccountLiquidity(address,address,uint256,uint256) (ComptrollerG5.sol#2601-2608)
_setPriceOracle(PriceOracle) should be declared external:
        - Comptroller._setPriceOracle(PriceOracle) (ComptrollerG5.sol#2710-2722)
_setPauseGuardian(address) should be declared external:
        - Comptroller._setPauseGuardian(address) (ComptrollerG5.sol#2858-2870)
```

```
_setPauseGuardian(address) should be declared external:
        - Comptroller._setPauseGuardian(address) (ComptrollerG5.sol#2858-2870)
_setMintPaused(YToken,bool) should be declared external:
        - Comptroller._setMintPaused(YToken,bool) (ComptrollerG5.sol#2872-2880)
_setBorrowPaused(YToken,bool) should be declared external:
        - Comptroller._setBorrowPaused(YToken,bool) (ComptrollerG5.sol#2882-2890)
_setTransferPaused(bool) should be declared external:
        - Comptroller._setTransferPaused(bool) (ComptrollerG5.sol#2892-2899)
_setSeizePaused(bool) should be declared external:
        - Comptroller._setSeizePaused(bool) (ComptrollerG5.sol#2901-2908)
_become(Unitroller) should be declared external:
        - Comptroller._become(Unitroller) (ComptrollerG5.sol#2910-2913)
_setLoanSpeeds(YToken[],uint256[],uint256[]) should be declared external:
        - Comptroller._setLoanSpeeds(YToken[],uint256[],uint256[]) (ComptrollerG5.sol#2920-2929)
claimLoan(address) should be declared external:
        - Comptroller.claimLoan(address) (ComptrollerG5.sol#3035-3037)
_setContributorLoanSpeed(address,uint256) should be declared external:
        - Comptroller._setContributorLoanSpeed(address,uint256) (ComptrollerG5.sol#3094-3107)
_grantLOAN(address,uint256) should be declared external:
        - Comptroller._grantLOAN(address,uint256) (ComptrollerG5.sol#3109-3116)
_setLoanRate(uint256) should be declared external:
        - Comptroller._setLoanRate(uint256) (ComptrollerG5.sol#3118-3125)
_dropLoanMarket(address) should be declared external:
        - Comptroller._dropLoanMarket(address) (ComptrollerG5.sol#3127-3136)
getAllMarkets() should be declared external:
        - Comptroller.getAllMarkets() (ComptrollerG5.sol#3138-3140)
canClaimLoanBySuppling(address) should be declared external:
        - Comptroller.canClaimLoanBySuppling(address) (ComptrollerG5.sol#3151-3160)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ComptrollerG5.sol analyzed (28 contracts with 75 detectors), 275 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> ComptrollerInterface.sol

```
INFO:Detectors:
Constant ComptrollerInterface.isComptroller (ComptrollerInterface.sol#5) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Slither:ComptrollerInterface.sol analyzed (2 contracts with 75 detectors), 1 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> ComptrollerStorage.sol

```
INFO:Detectors:
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (ComptrollerStorage.sol#2604-2612)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (ComptrollerStorage.sol#1111)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (ComptrollerStorage.sol#1158)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (ComptrollerStorage.sol#1167)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (ComptrollerStorage.sol#1172)
initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - YToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (ComptrollerStorage.sol#1185-1
216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ComptrollerStorage.sol analyzed (25 contracts with 75 detectors), 163 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> DAIInterestRateModelV3.sol

```
INFO:Detectors:
DAIInterestRateModelV3.updateJumpRateModel(uint256,uint256,uint256,uint256) (DAIInterestRateModelV3.sol#248-253) should emit a
n event for:
        - gapPerBlock = gapPerYear / blocksPerYear (DAIInterestRateModelV3.sol#250)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
JumpRateModelV2.constructor(uint256,uint256,uint256,uint256,address).owner_ (DAIInterestRateModelV3.sol#124) lacks a zero-chec
k on :
                - owner = owner_ (DAIInterestRateModelV3.sol#125)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
SafeMath.add(uint256,uint256,string) (DAIInterestRateModelV3.sol#20-25) is never used and should be removed
SafeMath.mod(uint256,uint256) (DAIInterestRateModelV3.sol#71-73) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (DAIInterestRateModelV3.sol#75-78) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (DAIInterestRateModelV3.sol#49-58) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Constant InterestRateModel.isInterestRateModel (DAIInterestRateModelV3.sol#4) is not in UPPER_CASE_WITH_UNDERSCORES
Constant JumpRateModelV2.blocksPerYear (DAIInterestRateModelV3.sol#94) is not in UPPER_CASE_WITH_UNDERSCORES
Constant DAIInterestRateModelV3.assumedOneMinusReserveFactorMantissa (DAIInterestRateModelV3.sol#221) is not in UPPER_CASE_WIT
H_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
JugLike.base (DAIInterestRateModelV3.sol#328) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Slither:DAIInterestRateModelV3.sol analyzed (6 contracts with 75 detectors), 13 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> ErrorReporter.sol

```
INFO:Detectors:
ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (ErrorReporter.sol#58-62) i
s never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (ErrorReporte
r.sol#67-71) is never used and should be removed
TokenErrorReporter.fail(TokenErrorReporter.Error,TokenErrorReporter.FailureInfo) (ErrorReporter.sol#199-203) is never used and
 should be removed
TokenErrorReporter.failOpaque(TokenErrorReporter.Error,TokenErrorReporter.FailureInfo,uint256) (ErrorReporter.sol#208-212) is
never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Slither:ErrorReporter.sol analyzed (2 contracts with 75 detectors), 4 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Exponential.sol

```
INFO:Detectors:
Exponential.divScalarByExpTruncate(uint256,Exponential.Exp).fraction (Exponential.sol#210) shadows:
        - Exponential.fraction(uint256,uint256) (Exponential.sol#422-424) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

```
INFO:Detectors:
Constant Exponential.expScale (Exponential.sol#88) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.doubleScale (Exponential.sol#89) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.halfExpScale (Exponential.sol#90) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.mantissaOne (Exponential.sol#91) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Exponential.mantissaOne (Exponential.sol#91) is never used in Exponential (Exponential.sol#87-425)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Slither:Exponential.sol analyzed (2 contracts with 75 detectors), 56 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> InterestRateModel.sol

```
INFO:Detectors:
Constant InterestRateModel.isInterestRateModel (InterestRateModel.sol#3) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Slither:InterestRateModel.sol analyzed (1 contracts with 75 detectors), 1 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> JumpRateModel.sol

```
INFO:Detectors:
SafeMath.add(uint256,uint256,string) (JumpRateModel.sol#11-16) is never used and should be removed
SafeMath.mod(uint256,uint256) (JumpRateModel.sol#62-64) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (JumpRateModel.sol#66-69) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (JumpRateModel.sol#40-49) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Constant InterestRateModel.isInterestRateModel (JumpRateModel.sol#73) is not in UPPER_CASE_WITH_UNDERSCORES
Constant JumpRateModel.blocksPerYear (JumpRateModel.sol#89) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
getSupplyRate(uint256,uint256,uint256,uint256) should be declared external:
        - JumpRateModel.getSupplyRate(uint256,uint256,uint256,uint256) (JumpRateModel.sol#170-175)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:JumpRateModel.sol analyzed (3 contracts with 75 detectors), 8 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> JumpRateModelV2.sol

```
INFO:Detectors:
JumpRateModelV2.constructor(uint256,uint256,uint256,uint256,address).owner_ (JumpRateModelV2.sol#123) lacks a zero-check on :
                - owner = owner_ (JumpRateModelV2.sol#124)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
SafeMath.add(uint256,uint256,string) (JumpRateModelV2.sol#19-24) is never used and should be removed
SafeMath.mod(uint256,uint256) (JumpRateModelV2.sol#70-72) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (JumpRateModelV2.sol#74-77) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (JumpRateModelV2.sol#48-57) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Constant InterestRateModel.isInterestRateModel (JumpRateModelV2.sol#3) is not in UPPER_CASE_WITH_UNDERSCORES
Constant JumpRateModelV2.blocksPerYear (JumpRateModelV2.sol#93) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
getSupplyRate(uint256,uint256,uint256,uint256) should be declared external:
        - JumpRateModelV2.getSupplyRate(uint256,uint256,uint256,uint256) (JumpRateModelV2.sol#185-190)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:JumpRateModelV2.sol analyzed (3 contracts with 75 detectors), 9 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> LoanAggregatorPriceOracle.sol

```
INFO:Detectors:
SDelegationStorage.implementation (LoanAggregatorPriceOracle.sol#1260) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (LoanAggregatorPriceOracle.sol#2721-2729)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (LoanAggregatorPriceOracle.sol#1228)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (LoanAggregatorPriceOracle.sol#1275)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (LoanAggregatorPriceOracle.sol#1284)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (LoanAggregatorPriceOracle.sol#1289)
initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - YErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (LoanAggregatorPriceOr
acle.sol#2814-2827)
getUnderlyingPrice(YToken) should be declared external:
        - LoanAggregatorPriceOracle.getUnderlyingPrice(YToken) (LoanAggregatorPriceOracle.sol#3038-3074)
setUnderlyingPrice(YToken,uint256) should be declared external:
        - LoanAggregatorPriceOracle.setUnderlyingPrice(YToken,uint256) (LoanAggregatorPriceOracle.sol#3076-3081)
setDirectPrice(address,uint256) should be declared external:
        - LoanAggregatorPriceOracle.setDirectPrice(address,uint256) (LoanAggregatorPriceOracle.sol#3083-3087)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:LoanAggregatorPriceOracle.sol analyzed (23 contracts with 75 detectors), 144 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> LoanPriceOracle.sol

```
INFO:Detectors:
SDelegationStorage.implementation (LoanPriceOracle.sol#1211) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (LoanPriceOracle.sol#2672-2680)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (LoanPriceOracle.sol#1179)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (LoanPriceOracle.sol#1226)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (LoanPriceOracle.sol#1235)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (LoanPriceOracle.sol#1240)
initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - YErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (LoanPriceOracle.sol#2
765-2778)
setRefAddress(IStdReference) should be declared external:
        - LoanPriceOracle.setRefAddress(IStdReference) (LoanPriceOracle.sol#2986-2989)
getRefAddress() should be declared external:
        - LoanPriceOracle.getRefAddress() (LoanPriceOracle.sol#2991-2993)
getUnderlyingPrice(YToken) should be declared external:
        - LoanPriceOracle.getUnderlyingPrice(YToken) (LoanPriceOracle.sol#2995-3015)
setUnderlyingPrice(YToken,uint256) should be declared external:
        - LoanPriceOracle.setUnderlyingPrice(YToken,uint256) (LoanPriceOracle.sol#3017-3022)
setDirectPrice(address,uint256) should be declared external:
        - LoanPriceOracle.setDirectPrice(address,uint256) (LoanPriceOracle.sol#3024-3028)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:LoanPriceOracle.sol analyzed (22 contracts with 75 detectors), 150 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Maximillion.sol

```
INFO:Detectors:
Variable YTokenInterface.seize(address,address,uint256).seizeTokens (Maximillion.sol#1101) is too similar to YToken.seizeInter
(address,address,address,uint256).seizerToken (Maximillion.sol#2216)
Variable YToken.seize(address,address,uint256).seizeTokens (Maximillion.sol#2190) is too similar to YToken.seizeInternal(addre
address,address,uint256).seizerToken (Maximillion.sol#2216)
Variable YToken.liquidateBorrowFresh(address,address,uint256,YTokenInterface).seizeTokens (Maximillion.sol#2155) is too simila
o YToken.seizeInternal(address,address,address,uint256).seizerToken (Maximillion.sol#2216)
Variable YToken.seizeInternal(address,address,address,uint256).seizeTokens (Maximillion.sol#2216) is too similar to YToken.sei
nternal(address,address,address,uint256).seizerToken (Maximillion.sol#2216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

```
INFO:Detectors:
SDelegationStorage.implementation (Maximillion.sol#1143) should be constant
YErc20Storage.underlying (Maximillion.sol#1118) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (Maximillion.sol#2604-2612)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (Maximillion.sol#1111)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (Maximillion.sol#1158)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (Maximillion.sol#1167)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (Maximillion.sol#1172)
repayBehalf(address) should be declared external:
        - Maximillion.repayBehalf(address) (Maximillion.sol#2864-2866)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Maximillion.sol analyzed (19 contracts with 75 detectors), 137 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> PriceOracle.sol

```
INFO:Detectors:
YToken (PriceOracle.sol#1175-2684) does not implement functions:
        - YToken.doTransferIn(address,uint256) (PriceOracle.sol#2663)
        - YToken.doTransferOut(address,uint256) (PriceOracle.sol#2670)
        - YToken.getCashPrior() (PriceOracle.sol#2657)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
SDelegationStorage.implementation (PriceOracle.sol#1143) should be constant
YErc20Storage.underlying (PriceOracle.sol#1118) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (PriceOracle.sol#2604-2612)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (PriceOracle.sol#1111)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (PriceOracle.sol#1158)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (PriceOracle.sol#1167)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (PriceOracle.sol#1172)
initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - YToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (PriceOracle.sol#1185-1216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:PriceOracle.sol analyzed (18 contracts with 75 detectors), 146 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Reservoir.sol

```
INFO:Detectors:
Reservoir.constructor(uint256,EIP20Interface,address).target_ (Reservoir.sol#90) lacks a zero-check on :
                - target = target_ (Reservoir.sol#94)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
drip() should be declared external:
        - Reservoir.drip() (Reservoir.sol#103-124)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Reservoir.sol analyzed (2 contracts with 75 detectors), 4 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> SimplePriceOracle.sol

```
INFO:Detectors:
SDelegationStorage.implementation (SimplePriceOracle.sol#1142) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (SimplePriceOracle.sol#2603-2611)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (SimplePriceOracle.sol#1110)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (SimplePriceOracle.sol#1157)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (SimplePriceOracle.sol#1166)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (SimplePriceOracle.sol#1171)
initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - YErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (SimplePriceOracle.sol
16-2729)
getUnderlyingPrice(YToken) should be declared external:
        - SimplePriceOracle.getUnderlyingPrice(YToken) (SimplePriceOracle.sol#2899-2905)
setUnderlyingPrice(YToken,uint256) should be declared external:
        - SimplePriceOracle.setUnderlyingPrice(YToken,uint256) (SimplePriceOracle.sol#2907-2911)
setDirectPrice(address,uint256) should be declared external:
        - SimplePriceOracle.setDirectPrice(address,uint256) (SimplePriceOracle.sol#2913-2916)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SimplePriceOracle.sol analyzed (20 contracts with 75 detectors), 137 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Timelock.sol

```
INFO:Detectors:
Low level call in Timelock.executeTransaction(address,uint256,string,bytes,uint256) (Timelock.sol#147-172):
        - (success,returnData) = target.call.value(value)(callData) (Timelock.sol#166)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
INFO:Detectors:
setDelay(uint256) should be declared external:
        - Timelock.setDelay(uint256) (Timelock.sol#103-110)
acceptAdmin() should be declared external:
        - Timelock.acceptAdmin() (Timelock.sol#112-118)
setPendingAdmin(address) should be declared external:
        - Timelock.setPendingAdmin(address) (Timelock.sol#120-125)
queueTransaction(address,uint256,string,bytes,uint256) should be declared external:
        - Timelock.queueTransaction(address,uint256,string,bytes,uint256) (Timelock.sol#127-136)
cancelTransaction(address,uint256,string,bytes,uint256) should be declared external:
        - Timelock.cancelTransaction(address,uint256,string,bytes,uint256) (Timelock.sol#138-145)
executeTransaction(address,uint256,string,bytes,uint256) should be declared external:
        - Timelock.executeTransaction(address,uint256,string,bytes,uint256) (Timelock.sol#147-172)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Timelock.sol analyzed (2 contracts with 75 detectors), 22 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Unitroller.sol

```
INFO:Detectors:
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (Unitroller.sol#2604-2612)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (Unitroller.sol#1111)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (Unitroller.sol#1158)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (Unitroller.sol#1167)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (Unitroller.sol#1172)
initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - YToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (Unitroller.sol#1185-1216)
_setPendingImplementation(address) should be declared external:
        - Unitroller._setPendingImplementation(address) (Unitroller.sol#2879-2892)
_acceptImplementation() should be declared external:
        - Unitroller._acceptImplementation() (Unitroller.sol#2899-2917)
_setPendingAdmin(address) should be declared external:
        - Unitroller._setPendingAdmin(address) (Unitroller.sol#2926-2942)
_acceptAdmin() should be declared external:
        - Unitroller._acceptAdmin() (Unitroller.sol#2949-2969)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Unitroller.sol analyzed (26 contracts with 75 detectors), 171 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> WhitePaperInterestRateModel.sol

```
INFO:Detectors:
SafeMath.add(uint256,uint256,string) (WhitePaperInterestRateModel.sol#11-16) is never used and should be removed
SafeMath.mod(uint256,uint256) (WhitePaperInterestRateModel.sol#62-64) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (WhitePaperInterestRateModel.sol#66-69) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (WhitePaperInterestRateModel.sol#40-49) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Constant InterestRateModel.isInterestRateModel (WhitePaperInterestRateModel.sol#73) is not in UPPER_CASE_WITH_UNDERSCORES
Constant WhitePaperInterestRateModel.blocksPerYear (WhitePaperInterestRateModel.sol#89) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
getSupplyRate(uint256,uint256,uint256,uint256) should be declared external:
        - WhitePaperInterestRateModel.getSupplyRate(uint256,uint256,uint256,uint256) (WhitePaperInterestRateModel.sol#149-154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:WhitePaperInterestRateModel.sol analyzed (3 contracts with 75 detectors), 8 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> YDaiDelegate.sol

```
INFO:Detectors:
Redundant expression "data (YDaiDelegate.sol#2893)" inYErc20Delegate (YDaiDelegate.sol#2881-2914)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable YToken.seizeInternal(address,address,address,uint256).seizeTokens (YDaiDelegate.sol#2216) is too similar to YToken.se
Internal(address,address,address,uint256).seizerToken (YDaiDelegate.sol#2216)
Variable YTokenInterface.seize(address,address,uint256).seizeTokens (YDaiDelegate.sol#1101) is too similar to YToken.seizeInte
l(address,address,address,uint256).seizerToken (YDaiDelegate.sol#2216)
Variable YToken.seize(address,address,uint256).seizeTokens (YDaiDelegate.sol#2190) is too similar to YToken.seizeInternal(addr
,address,address,uint256).seizerToken (YDaiDelegate.sol#2216)
Variable YToken.liquidateBorrowFresh(address,address,uint256,YTokenInterface).seizeTokens (YDaiDelegate.sol#2155) is too simil
to YToken.seizeInternal(address,address,address,uint256).seizerToken (YDaiDelegate.sol#2216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (YDaiDelegate.sol#2604-2612)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (YDaiDelegate.sol#1111)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (YDaiDelegate.sol#1158)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (YDaiDelegate.sol#1167)
        - YDaiDelegate._becomeImplementation(bytes) (YDaiDelegate.sol#2936-2941)
        - YErc20Delegate._becomeImplementation(bytes) (YDaiDelegate.sol#2891-2901)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (YDaiDelegate.sol#1172)
        - YDaiDelegate._resignImplementation() (YDaiDelegate.sol#2978-2998)
        - YErc20Delegate._resignImplementation() (YDaiDelegate.sol#2906-2913)
initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - YErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (YDaiDelegate.sol#2702
15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:YDaiDelegate.sol analyzed (24 contracts with 75 detectors), 150 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> YErc20.sol

```
INFO:Detectors:
SDelegationStorage.implementation (YErc20.sol#1143) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (YErc20.sol#2604-2612)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (YErc20.sol#1111)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (YErc20.sol#1158)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (YErc20.sol#1167)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (YErc20.sol#1172)
initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - YErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (YErc20.sol#2702-2715)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:YErc20.sol analyzed (18 contracts with 75 detectors), 133 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> YErc20Delegate.sol

```
INFO:Detectors:
Redundant expression "data (YErc20Delegate.sol#2893)" inYErc20Delegate (YErc20Delegate.sol#2881-2914)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable YToken.seize(address,address,uint256).seizeTokens (YErc20Delegate.sol#2190) is too similar to YToken.seizeInternal(ad
ss,address,address,uint256).seizerToken (YErc20Delegate.sol#2216)
Variable YToken.liquidateBorrowFresh(address,address,uint256,YTokenInterface).seizeTokens (YErc20Delegate.sol#2155) is too sim
r to YToken.seizeInternal(address,address,address,uint256).seizerToken (YErc20Delegate.sol#2216)
Variable YToken.seizeInternal(address,address,address,uint256).seizeTokens (YErc20Delegate.sol#2216) is too similar to YToken.
zeInternal(address,address,address,uint256).seizerToken (YErc20Delegate.sol#2216)
Variable YTokenInterface.seize(address,address,uint256).seizeTokens (YErc20Delegate.sol#1101) is too similar to YToken.seizeIn
nal(address,address,address,uint256).seizerToken (YErc20Delegate.sol#2216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (YErc20Delegate.sol#2604-2612)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (YErc20Delegate.sol#1111)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (YErc20Delegate.sol#1158)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (YErc20Delegate.sol#1167)
        - YErc20Delegate._becomeImplementation(bytes) (YErc20Delegate.sol#2891-2901)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (YErc20Delegate.sol#1172)
        - YErc20Delegate._resignImplementation() (YErc20Delegate.sol#2906-2913)
initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - YErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (YErc20Delegate.sol#27
2715)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:YErc20Delegate.sol analyzed (19 contracts with 75 detectors), 137 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> YErc20Delegator.sol

```
INFO:Detectors:
borrowBalanceStored(address) should be declared external:
        - YErc20Delegator.borrowBalanceStored(address) (YErc20Delegator.sol#678-681)
        - YTokenInterface.borrowBalanceStored(address) (YErc20Delegator.sol#336)
exchangeRateCurrent() should be declared external:
        - YErc20Delegator.exchangeRateCurrent() (YErc20Delegator.sol#687-690)
        - YTokenInterface.exchangeRateCurrent() (YErc20Delegator.sol#337)
exchangeRateStored() should be declared external:
        - YErc20Delegator.exchangeRateStored() (YErc20Delegator.sol#697-700)
        - YTokenInterface.exchangeRateStored() (YErc20Delegator.sol#338)
accrueInterest() should be declared external:
        - YErc20Delegator.accrueInterest() (YErc20Delegator.sol#716-719)
        - YTokenInterface.accrueInterest() (YErc20Delegator.sol#340)
_setComptroller(ComptrollerInterface) should be declared external:
        - YErc20Delegator._setComptroller(ComptrollerInterface) (YErc20Delegator.sol#753-756)
        - YTokenInterface._setComptroller(ComptrollerInterface) (YErc20Delegator.sol#348)
_setInterestRateModel(InterestRateModel) should be declared external:
        - YErc20Delegator._setInterestRateModel(InterestRateModel) (YErc20Delegator.sol#814-817)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (YErc20Delegator.sol#351)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (YErc20Delegator.sol#407)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (YErc20Delegator.sol#412)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:YErc20Delegator.sol analyzed (11 contracts with 75 detectors), 68 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> YErc20Immutable.sol

```
INFO:Detectors:
SDelegationStorage.implementation (YErc20Immutable.sol#1143) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (YErc20Immutable.sol#2604-2612)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (YErc20Immutable.sol#1111)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (YErc20Immutable.sol#1158)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (YErc20Immutable.sol#1167)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (YErc20Immutable.sol#1172)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:YErc20Immutable.sol analyzed (19 contracts with 75 detectors), 133 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> YEther.sol

```
INFO:Detectors:
Variable YToken.liquidateBorrowFresh(address,address,uint256,YTokenInterface).seizeTokens (YEther.sol#2155) is too similar to
ken.seizeInternal(address,address,address,uint256).seizerToken (YEther.sol#2216)
Variable YToken.seize(address,address,uint256).seizeTokens (YEther.sol#2190) is too similar to YToken.seizeInternal(address,ad
ss,address,uint256).seizerToken (YEther.sol#2216)
Variable YTokenInterface.seize(address,address,uint256).seizeTokens (YEther.sol#1101) is too similar to YToken.seizeInternal(a
ess,address,address,uint256).seizerToken (YEther.sol#2216)
Variable YToken.seizeInternal(address,address,address,uint256).seizeTokens (YEther.sol#2216) is too similar to YToken.seizeInt
al(address,address,address,uint256).seizerToken (YEther.sol#2216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
SDelegationStorage.implementation (YEther.sol#1143) should be constant
YErc20Storage.underlying (YEther.sol#1118) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (YEther.sol#2604-2612)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (YEther.sol#1111)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (YEther.sol#1158)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (YEther.sol#1167)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (YEther.sol#1172)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:YEther.sol analyzed (18 contracts with 75 detectors), 135 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> YLoanLikeDelegate.sol

```
INFO:Detectors:
Variable YToken.seize(address,address,uint256).seizeTokens (YLoanLikeDelegate.sol#2190) is too similar to YToken.seizeInternal(
dress,address,address,uint256).seizerToken (YLoanLikeDelegate.sol#2216)
Variable YToken.seizeInternal(address,address,address,uint256).seizeTokens (YLoanLikeDelegate.sol#2216) is too similar to YToke
seizeInternal(address,address,address,uint256).seizerToken (YLoanLikeDelegate.sol#2216)
Variable YTokenInterface.seize(address,address,uint256).seizeTokens (YLoanLikeDelegate.sol#1101) is too similar to YToken.seize
ternal(address,address,address,uint256).seizerToken (YLoanLikeDelegate.sol#2216)
Variable YToken.liquidateBorrowFresh(address,address,uint256,YTokenInterface).seizeTokens (YLoanLikeDelegate.sol#2155) is too s
ilar to YToken.seizeInternal(address,address,address,uint256).seizerToken (YLoanLikeDelegate.sol#2216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (YLoanLikeDelegate.sol#2604-2612)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (YLoanLikeDelegate.sol#1111)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (YLoanLikeDelegate.sol#1158)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (YLoanLikeDelegate.sol#1167)
        - YErc20Delegate._becomeImplementation(bytes) (YLoanLikeDelegate.sol#2891-2901)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (YLoanLikeDelegate.sol#1172)
        - YErc20Delegate._resignImplementation() (YLoanLikeDelegate.sol#2906-2913)
initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - YErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (YLoanLikeDelegate.sol#
02-2715)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:YLoanLikeDelegate.sol analyzed (21 contracts with 75 detectors), 138 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> YToken.sol

```
INFO:Detectors:
Exponential.divScalarByExpTruncate(uint256,Exponential.Exp).fraction (YToken.sol#334) shadows:
        - Exponential.fraction(uint256,uint256) (YToken.sol#546-548) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
YToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (YToken.sol#1185-1216) should emit an eve
 for:
        - initialExchangeRateMantissa = initialExchangeRateMantissa_ (YToken.sol#1195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
YToken._setPendingAdmin(address).newPendingAdmin (YToken.sol#2294) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (YToken.sol#2304)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
INFO:Detectors:
YToken (YToken.sol#1175-2684) does not implement functions:
        - YToken.doTransferIn(address,uint256) (YToken.sol#2663)
        - YToken.doTransferOut(address,uint256) (YToken.sol#2670)
        - YToken.getCashPrior() (YToken.sol#2657)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
SDelegationStorage.implementation (YToken.sol#1143) should be constant
YErc20Storage.underlying (YToken.sol#1118) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (YToken.sol#2604-2612)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (YToken.sol#1111)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (YToken.sol#1158)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (YToken.sol#1167)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (YToken.sol#1172)
initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - YToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (YToken.sol#1185-1216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:YToken.sol analyzed (17 contracts with 75 detectors), 145 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> YTokenInterfaces.sol

```
INFO:Detectors:
borrowBalanceStored(address) should be declared external:
        - YTokenInterface.borrowBalanceStored(address) (YTokenInterfaces.sol#336)
exchangeRateCurrent() should be declared external:
        - YTokenInterface.exchangeRateCurrent() (YTokenInterfaces.sol#337)
exchangeRateStored() should be declared external:
        - YTokenInterface.exchangeRateStored() (YTokenInterfaces.sol#338)
accrueInterest() should be declared external:
        - YTokenInterface.accrueInterest() (YTokenInterfaces.sol#340)
_setComptroller(ComptrollerInterface) should be declared external:
        - YTokenInterface._setComptroller(ComptrollerInterface) (YTokenInterfaces.sol#348)
_setInterestRateModel(InterestRateModel) should be declared external:
        - YTokenInterface._setInterestRateModel(InterestRateModel) (YTokenInterfaces.sol#351)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (YTokenInterfaces.sol#398)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (YTokenInterfaces.sol#407)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (YTokenInterfaces.sol#412)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:YTokenInterfaces.sol analyzed (10 contracts with 75 detectors), 41 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> GovernorAlpha.sol

```
INFO:Detectors:
GovernorAlpha.quorumVotes() (GovernorAlpha.sol#9) uses literals with too many digits:
        - 8000000e18 (GovernorAlpha.sol#9)
GovernorAlpha.proposalThreshold() (GovernorAlpha.sol#12) uses literals with too many digits:
        - 2000000e18 (GovernorAlpha.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
propose(address[],uint256[],string[],bytes[],string) should be declared external:
        - GovernorAlpha.propose(address[],uint256[],string[],bytes[],string) (GovernorAlpha.sol#136-174)
queue(uint256) should be declared external:
        - GovernorAlpha.queue(uint256) (GovernorAlpha.sol#176-185)
execute(uint256) should be declared external:
        - GovernorAlpha.execute(uint256) (GovernorAlpha.sol#192-200)
cancel(uint256) should be declared external:
        - GovernorAlpha.cancel(uint256) (GovernorAlpha.sol#202-215)
getActions(uint256) should be declared external:
        - GovernorAlpha.getActions(uint256) (GovernorAlpha.sol#217-220)
getReceipt(uint256,address) should be declared external:
        - GovernorAlpha.getReceipt(uint256,address) (GovernorAlpha.sol#222-224)
castVote(uint256,bool) should be declared external:
        - GovernorAlpha.castVote(uint256,bool) (GovernorAlpha.sol#248-250)
castVoteBySig(uint256,bool,uint8,bytes32,bytes32) should be declared external:
        - GovernorAlpha.castVoteBySig(uint256,bool,uint8,bytes32,bytes32) (GovernorAlpha.sol#252-259)
__acceptAdmin() should be declared external:
        - GovernorAlpha.__acceptAdmin() (GovernorAlpha.sol#281-284)
__abdicate() should be declared external:
        - GovernorAlpha.__abdicate() (GovernorAlpha.sol#286-289)
__queueSetTimelockPendingAdmin(address,uint256) should be declared external:
        - GovernorAlpha.__queueSetTimelockPendingAdmin(address,uint256) (GovernorAlpha.sol#291-294)
__executeSetTimelockPendingAdmin(address,uint256) should be declared external:
        - GovernorAlpha.__executeSetTimelockPendingAdmin(address,uint256) (GovernorAlpha.sol#296-299)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:GovernorAlpha.sol analyzed (3 contracts with 75 detectors), 33 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> LOAN.sol

```
INFO:Detectors:
LOAN.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (LOAN.sol#161-170) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,Loan::delegateBySig: signature expired) (LOAN.sol#168)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
LOAN.getChainId() (LOAN.sol#296-300) uses assembly
        - INLINE ASM (LOAN.sol#298)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Constant LOAN.totalSupply (LOAN.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
LOAN.slitherConstructorConstantVariables() (LOAN.sol#4-301) uses literals with too many digits:
        - totalSupply = 200000000e18 (LOAN.sol#15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
delegate(address) should be declared external:
        - LOAN.delegate(address) (LOAN.sol#148-150)
delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - LOAN.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (LOAN.sol#161-170)
getPriorVotes(address,uint256) should be declared external:
        - LOAN.getPriorVotes(address,uint256) (LOAN.sol#189-221)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:LOAN.sol analyzed (1 contracts with 75 detectors), 8 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> LoanLens.sol

```
INFO:Detectors:
LOAN.slitherConstructorConstantVariables() (LoanLens.sol#5-302) uses literals with too many digits:
        - totalSupply = 200000000e18 (LoanLens.sol#16)
GovernorAlpha.quorumVotes() (LoanLens.sol#309) uses literals with too many digits:
        - 8000000e18 (LoanLens.sol#309)
GovernorAlpha.proposalThreshold() (LoanLens.sol#312) uses literals with too many digits:
        - 2000000e18 (LoanLens.sol#312)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
GovernorAlpha.proposalCount (LoanLens.sol#333) should be constant
SDelegationStorage.implementation (LoanLens.sol#1735) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
delegate(address) should be declared external:
        - LOAN.delegate(address) (LoanLens.sol#149-151)
delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - LOAN.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (LoanLens.sol#162-171)
getPriorVotes(address,uint256) should be declared external:
        - LOAN.getPriorVotes(address,uint256) (LoanLens.sol#190-222)
proposalMaxOperations() should be declared external:
        - GovernorAlpha.proposalMaxOperations() (LoanLens.sol#315)
votingDelay() should be declared external:
        - GovernorAlpha.votingDelay() (LoanLens.sol#318)
votingPeriod() should be declared external:
        - GovernorAlpha.votingPeriod() (LoanLens.sol#321)
queue(uint256) should be declared external:
        - GovernorAlpha.queue(uint256) (LoanLens.sol#438-447)
execute(uint256) should be declared external:
        - GovernorAlpha.execute(uint256) (LoanLens.sol#454-462)
cancel(uint256) should be declared external:
        - GovernorAlpha.cancel(uint256) (LoanLens.sol#464-477)
getActions(uint256) should be declared external:
        - GovernorAlpha.getActions(uint256) (LoanLens.sol#479-482)
getReceipt(uint256,address) should be declared external:
        - GovernorAlpha.getReceipt(uint256,address) (LoanLens.sol#484-486)
```

```
getActions(uint256) should be declared external:
        - GovernorAlpha.getActions(uint256) (LoanLens.sol#479-482)
getReceipt(uint256,address) should be declared external:
        - GovernorAlpha.getReceipt(uint256,address) (LoanLens.sol#484-486)
castVote(uint256,bool) should be declared external:
        - GovernorAlpha.castVote(uint256,bool) (LoanLens.sol#510-512)
castVoteBySig(uint256,bool,uint8,bytes32,bytes32) should be declared external:
        - GovernorAlpha.castVoteBySig(uint256,bool,uint8,bytes32,bytes32) (LoanLens.sol#514-521)
__acceptAdmin() should be declared external:
        - GovernorAlpha.__acceptAdmin() (LoanLens.sol#543-546)
__abdicate() should be declared external:
        - GovernorAlpha.__abdicate() (LoanLens.sol#548-551)
__queueSetTimelockPendingAdmin(address,uint256) should be declared external:
        - GovernorAlpha.__queueSetTimelockPendingAdmin(address,uint256) (LoanLens.sol#553-556)
__executeSetTimelockPendingAdmin(address,uint256) should be declared external:
        - GovernorAlpha.__executeSetTimelockPendingAdmin(address,uint256) (LoanLens.sol#558-561)
_setInterestRateModel(InterestRateModel) should be declared external:
        - YToken._setInterestRateModel(InterestRateModel) (LoanLens.sol#3196-3204)
        - YTokenInterface._setInterestRateModel(InterestRateModel) (LoanLens.sol#1703)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (LoanLens.sol#1750)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (LoanLens.sol#1759)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (LoanLens.sol#1764)
initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - YErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (LoanLens.sol#3294-3307
)
getGovReceipts(GovernorAlpha,address,uint256[]) should be declared external:
        - LoanLens.getGovReceipts(GovernorAlpha,address,uint256[]) (LoanLens.sol#3632-3645)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:LoanLens.sol analyzed (25 contracts with 75 detectors), 184 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> LoanReserve.sol

```
INFO:Detectors:
LoanReserve.setRewarder(address)._rewarder (LoanReserve.sol#462) lacks a zero-check on :
                - rewarder = _rewarder (LoanReserve.sol#464)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Address.verifyCallResult(bool,bytes,string) (LoanReserve.sol#273-293) uses assembly
        - INLINE ASM (LoanReserve.sol#285-288)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
LoanReserve.setPool(address) (LoanReserve.sol#468-472) compares to a boolean constant:
        -require(bool,string)(allowedPools[_pool] == false,Reserve::setPool: ALREADY_ALLOWED) (LoanReserve.sol#469)
LoanReserve.removePool(address) (LoanReserve.sol#474-478) compares to a boolean constant:
        -require(bool,string)(allowedPools[_pool] == true,Reserve::removePool: NOT_ALLOWED) (LoanReserve.sol#475)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

```
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (LoanReserve.sol#422-424)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (LoanReserve.sol#430-433)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:LoanReserve.sol analyzed (6 contracts with 75 detectors), 31 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> LoanStaking.sol

```
INFO:Detectors:
LoanStaking (LoanStaking.sol#1053-1612) does not implement functions:
        - Context._msgData() (LoanStaking.sol#969-971)
        - Context._msgSender() (LoanStaking.sol#965-967)
        - LoanStaking._notifyReward(address,uint256) (LoanStaking.sol#1458-1469)
        - LoanStaking.depositNFT(address,uint256,uint256) (LoanStaking.sol#1554-1572)
        - LoanStaking.earnedBalances(address) (LoanStaking.sol#1238-1252)
        - LoanStaking.emergencyWithdraw() (LoanStaking.sol#1417-1431)
        - LoanStaking.getBoost(address) (LoanStaking.sol#1517-1526)
        - LoanStaking.getReward() (LoanStaking.sol#1404-1414)
        - LoanStaking.getSlots(address) (LoanStaking.sol#1528-1539)
        - LoanStaking.getTokenIds(address) (LoanStaking.sol#1541-1552)
        - Ownable.initOwner(address) (LoanStaking.sol#1026-1029)
        - LoanStaking.lockedBalances(address) (LoanStaking.sol#1255-1280)
        - LoanStaking.mint(address,uint256) (LoanStaking.sol#1332-1353)
        - LoanStaking.notifyRewardAmount(address,uint256) (LoanStaking.sol#1471-1479)
        - Ownable.owner() (LoanStaking.sol#1031-1033)
        - LoanStaking.recoverERC20(address,uint256) (LoanStaking.sol#1482-1487)
        - Ownable.renounceOwnership() (LoanStaking.sol#1040-1043)
        - LoanStaking.setNftBoostRate(uint256) (LoanStaking.sol#1593-1597)
        - LoanStaking.setNftController(address) (LoanStaking.sol#1588-1591)
        - LoanStaking.stake(uint256,bool) (LoanStaking.sol#1307-1327)
        - LoanStaking.totalBalance(address) (LoanStaking.sol#1219-1221)
        - Ownable.transferOwnership(address) (LoanStaking.sol#1045-1049)
        - LoanStaking.unlockedBalance(address) (LoanStaking.sol#1224-1234)
        - LoanStaking.withdraw(uint256) (LoanStaking.sol#1358-1401)
        - LoanStaking.withdrawExpiredLocks() (LoanStaking.sol#1434-1454)
        - LoanStaking.withdrawNFT(uint256) (LoanStaking.sol#1574-1586)
        - LoanStaking.withdrawableBalance(address) (LoanStaking.sol#1283-1301)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
```

```
INFO:Detectors:
owner() should be declared external:
        - Ownable.owner() (LoanStaking.sol#1031-1033)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (LoanStaking.sol#1040-1043)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (LoanStaking.sol#1045-1049)
addReward(address,address) should be declared external:
        - LoanStaking.addReward(address,address) (LoanStaking.sol#1150-1158)
withdrawableBalance(address) should be declared external:
        - LoanStaking.withdrawableBalance(address) (LoanStaking.sol#1283-1301)
withdraw(uint256) should be declared external:
        - LoanStaking.withdraw(uint256) (LoanStaking.sol#1358-1401)
getReward() should be declared external:
        - LoanStaking.getReward() (LoanStaking.sol#1404-1414)
getBoost(address) should be declared external:
        - LoanStaking.getBoost(address) (LoanStaking.sol#1517-1526)
getSlots(address) should be declared external:
        - LoanStaking.getSlots(address) (LoanStaking.sol#1528-1539)
getTokenIds(address) should be declared external:
        - LoanStaking.getTokenIds(address) (LoanStaking.sol#1541-1552)
depositNFT(address,uint256,uint256) should be declared external:
        - LoanStaking.depositNFT(address,uint256,uint256) (LoanStaking.sol#1554-1572)
withdrawNFT(uint256) should be declared external:
        - LoanStaking.withdrawNFT(uint256) (LoanStaking.sol#1574-1586)
setNftController(address) should be declared external:
        - LoanStaking.setNftController(address) (LoanStaking.sol#1588-1591)
setNftBoostRate(uint256) should be declared external:
        - LoanStaking.setNftBoostRate(uint256) (LoanStaking.sol#1593-1597)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:LoanStaking.sol analyzed (13 contracts with 75 detectors), 95 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> LoanStakingProxy.sol

```
INFO:Detectors:
Pragma version^0.8.4 (LoanStakingProxy.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/
0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (LoanStakingProxy.sol#128-133):
        - (success) = recipient.call{value: amount}() (LoanStakingProxy.sol#131)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (LoanStakingProxy.sol#196-205):
        - (success,returndata) = target.call{value: value}(data) (LoanStakingProxy.sol#203)
Low level call in Address.functionStaticCall(address,bytes,string) (LoanStakingProxy.sol#223-230):
        - (success,returndata) = target.staticcall(data) (LoanStakingProxy.sol#228)
Low level call in Address.functionDelegateCall(address,bytes,string) (LoanStakingProxy.sol#248-255):
        - (success,returndata) = target.delegatecall(data) (LoanStakingProxy.sol#253)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Slither:LoanStakingProxy.sol analyzed (9 contracts with 75 detectors), 37 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> NFTController.sol

```
INFO:Detectors:
Pragma version0.8.4 (NFTController.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.
6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (NFTController.sol#48-50)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (NFTController.sol#56-59)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:NFTController.sol analyzed (3 contracts with 75 detectors), 5 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

CarefulMath.sol

Miscellaneous

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 26:12:

Comptroller.sol

Security

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 3285:39:

Gas & Economy

Gas costs:

Gas requirement of function Comptroller.updateContributorRewards is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 4658:22:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 4768:26:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 4377:26:

ComptrollerG1.sol

Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 4274:34:

Gas & Economy

Gas costs:

Gas requirement of function ComptrollerG1._setLiquidationIncentive is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 4185:22:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 4261:30:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 3492:24:

ComptrollerG2.sol

Security

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 3285:39:

Gas & Economy

Gas costs:

Gas requirement of function ComptrollerG2._setSeizePaused is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 4329:22:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 4342:26:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 3483:24:

ComptrollerG3.sol

Security

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 3285:39:

Gas & Economy

Gas costs:

Gas requirement of function ComptrollerG3.getAllMarkets is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 4666:22:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 4653:26:

ComptrollerG4.sol

Security

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.

more

Pos: 3285:39:

Gas & Economy

Gas costs:

Gas requirement of function ComptrollerG4.getAllMarkets is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 4675:22:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 4662:26:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.

more

Pos: 3482:24:

ComptrollerG5.sol

Security

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.

more

Pos: 3285:39:

Gas & Economy

Gas costs:

Gas requirement of function Comptroller.canClaimLoanBySuppling is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 4745:22:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 4394:26:

ComptrollerStorage.sol

Gas & Economy

Gas costs:

Gas requirement of function YTokenStorage.symbol is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 880:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 7:4:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 2679:16:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 543:15:

DAIInterestRateModelV3.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in DAIInterestRateModelV3.poke(): Could potentially lead to re-entrancy vulnerability.

more

Pos: 290:4:

Gas & Economy

Gas costs:

Gas requirement of function DAIInterestRateModelV3.poke is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 290:4:

Miscellaneous

Constant/View/Pure functions:

JumpRateModelV2.getSupplyRate(uint256,uint256,uint256,uint256) : Is constant but potentially should not be.

more

Pos: 187:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 250:8:

Exponential.sol

Miscellaneous

Constant/View/Pure functions:

Exponential.fraction(uint256,uint256) : Is constant but potentially should not be.

more

Pos: 422:4:

Email: audit@EtherAuthority.io

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 418:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 419:15:

JumpRateModel.sol

Gas & Economy

Gas costs:

Gas requirement of function JumpRateModel.getSupplyRate is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 170:4:

Miscellaneous

Constant/View/Pure functions:

JumpRateModel.getSupplyRate(uint256,uint256,uint256,uint256) : Is constant but potentially should not be.
more
Pos: 170:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 67:8:

JumpRateModelV2.sol

Gas & Economy

Gas costs:

Gas requirement of function JumpRateModelV2.getSupplyRate is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 185:4:

Miscellaneous

Constant/View/Pure functions:

JumpRateModelV2.getSupplyRate(uint256,uint256,uint256,uint256) : Is constant but potentially should not be.
more
Pos: 185:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 137:8:

LoanAggregatorPriceOracle.sol

Security

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 2976:16:

Gas & Economy

Gas costs:

Gas requirement of function LoanAggregatorPriceOracle.setAdmin is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 3124:12:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 3125:16:

LoanPriceOracle.sol

Security

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 2927:16:

Gas & Economy

Gas costs:

Gas requirement of function LoanPriceOracle.setAdmin is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 3038:12:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 77:4:

Miscellaneous

Similar variable names:

LoanPriceOracle.setDirectPrice(address,uint256) : Variables have very similar names "price" and "prices". Note: Modifiers are currently not considered by this static analysis.
Pos: 3028:0:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 3039:16:

Maximillion.sol

Gas & Economy

Gas costs:

Gas requirement of function Maximillion.repayBehalf is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 2864:12:

Miscellaneous

Similar variable names:

Maximillion.repayBehalfExplicit(address,contract YEther) : Variables have very similar names "borrows" and "borrower". Note: Modifiers are currently not considered by this static analysis.
Pos: 2881:62:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 2842:16:

String length:

"bytes" and "string" lengths are not the same since strings are assumed to be UTF-8 encoded (according to the ABI defintion) therefore one character is not nessesarily encoded in one byte of data.
more
Pos: 2829:53:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 2838:53:

PriceOracle.sol

Gas & Economy

Gas costs:

Gas requirement of function YTokenStorage.symbol is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 880:4:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 2679:16:

Reservoir.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Reservoir.drip(): Could potentially lead to re-entrancy vulnerability.
more
Pos: 103:2:

Gas & Economy

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 145:4:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 145:12:

SimplePriceOracle.sol

Security

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 2878:16:

Gas & Economy

Gas costs:

Gas requirement of function YErc20._addReserves is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 2812:12:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 2891:16:

Timelock.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Timelock.executeTransaction(address,uint256,string,bytes,uint256): Could potentially lead to re-entrancy vulnerability.
more
Pos: 147:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 176:15:

Gas & Economy

Gas costs:

Gas requirement of function Timelock.setDelay is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 103:4:

Miscellaneous

Constant/View/Pure functions:

SafeMath.sub(uint256,uint256) : Is constant but potentially should not be.
more
Pos: 17:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 167:8:

String length:

"bytes" and "string" lengths are not the same since strings are assumed to be UTF-8 encoded (according to the ABI defintion) therefore one character is not nessesarily encoded in one byte of data.
more
Pos: 159:12:

Unitroller.sol

Security

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 2978:39:

Gas & Economy

Gas costs:

Fallback function of contract Unitroller requires too much gas (infinite). If the fallback function requires more than 2300 gas, the contract cannot receive Ether.
Pos: 2976:16:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 2679:16:

WhitePaperInterestRateModel.sol

Gas & Economy

Gas costs:

Gas requirement of function WhitePaperInterestRateModel.getSupplyRate is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 149:4:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 67:8:

YDaiDelegate.sol

Security

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 2864:16:

Gas & Economy

Gas costs:

Gas requirement of function YErc20Delegate.accrueInterest is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 3007:12:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 3084:16:

YErc20.sol

Security

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 2864:16:

Gas & Economy

Gas costs:

Gas requirement of function YErc20._addReserves is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 2798:12:

Miscellaneous

Similar variable names:

YErc20.initialize(address,contract ComptrollerInterface,contract InterestRateModel,uint256,string,string,uint8) : Variables have very similar names "underlying" and "underlying_". Note: Modifiers are currently not considered by this static analysis.
Pos: 2714:31:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 2877:16:

YErc20Delegate.sol

Security

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 2864:16:

Gas & Economy

Gas costs:

Gas requirement of function YErc20Delegate._resignImplementation is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 2906:12:

Miscellaneous

Similar variable names:

YErc20.initialize(address,contract ComptrollerInterface,contract InterestRateModel,uint256,string,string,uint8) : Variables have very similar names "underlying" and "underlying_". Note: Modifiers are currently not considered by this static analysis.
Pos: 2714:31:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 2912:16:

YErc20Delegator.sol

Security

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 828:8:

Gas & Economy

Gas costs:

Fallback function of contract YErc20Delegator requires too much gas (infinite). If the fallback function requires more than 2300 gas, the contract cannot receive Ether.
Pos: 867:4:

Miscellaneous

Constant/View/Pure functions:

YErc20Delegator.delegateToViewImplementation(bytes) : Is constant but potentially should not be.
more
Pos: 853:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 868:8:

YErc20Immutable.sol

Security

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 2864:16:

Gas & Economy

Gas costs:

Gas requirement of function YErc20Immutable._addReserves is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 2798:12:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 2877:16:

YEther.sol

Gas & Economy

Gas costs:

Fallback function of contract YEther requires too much gas (infinite). If the fallback function requires more than 2300 gas, the contract cannot receive Ether.
Pos: 2788:12:

Miscellaneous

Similar variable names:

YEther.(contract ComptrollerInterface,contract InterestRateModel,uint256,string,string,uint8,address payable) : Variables have very similar names "initialExchangeRateMantissa" and "initialExchangeRateMantissa_". Note: Modifiers are currently not considered by this static analysis.
Pos: 2707:61:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 2842:16:

String length:

"bytes" and "string" lengths are not the same since strings are assumed to be UTF-8 encoded (according to the ABI defintion) therefore one character is not nessesarily encoded in one byte of data.
more
Pos: 2829:53:

YLoanLikeDelegate.sol

Security

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 2864:16:

Gas & Economy

Gas costs:

Gas requirement of function YLoanLikeDelegate._delegateLoanLikeTo is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 2935:10:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 2936:12:

YToken.sol

Gas & Economy

Gas costs:

Gas requirement of function YTokenStorage.symbol is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 880:4:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 2679:16:

YTokenInterfaces.sol

Gas & Economy

Gas costs:

Gas requirement of function YTokenStorage.name is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 115:4:

GovernorAlpha.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in GovernorAlpha._castVote(address,uint256,bool): Could potentially lead to re-entrancy vulnerability.

more

Pos: 261:4:

Gas & Economy

Gas costs:

Gas requirement of function GovernorAlpha.__executeSetTimelockPendingAdmin is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 296:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 196:8:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 308:8:

String length:

"bytes" and "string" lengths are not the same since strings are assumed to be UTF-8 encoded (according to the ABI defintion) therefore one character is not nessesarily encoded in one byte of data.
more
Pos: 138:108:

LOAN.sol

Security

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 298:8:

Block timestamp:

Use of "now": "now" does not mean current time. "now" is an alias for "block.timestamp". "block.timestamp" can be influenced by miners to a certain degree, be careful.
more
Pos: 168:16:

Gas & Economy

Gas costs:

Gas requirement of function LOAN.name is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 6:4:

Miscellaneous

Constant/View/Pure functions:

LOAN.getChainId() : Is constant but potentially should not be.
more
Pos: 296:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 292:8:

LoanLens.sol

Gas & Economy

Gas costs:

Gas requirement of function GovernorAlpha.name is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 7:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 3759:16:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 3779:16:

LoanReserve.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeDecreaseAllowance(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 346:4:

Gas & Economy

Gas costs:

Gas requirement of function LoanReserve.transfer is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 480:4:

Miscellaneous

Constant/View/Pure functions:

LoanReserve.transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 480:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 482:8:

LoanStaking.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in LoanStaking.withdrawNFT(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1574:6:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 1468:51:

Gas & Economy

Gas costs:

Gas requirement of function LoanStaking.setNftBoostRate is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 1593:6:

Delete dynamic array:

The "delete" operation when applied to a dynamically sized array in Solidity generates code to delete each of the elements contained. If the array is large, this operation can surpass the block gas limit and raise an OOG exception. Also nested dynamically sized objects can produce the same results.

more

Pos: 1441:14:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.

more

Pos: 1505:10:

Miscellaneous

Constant/View/Pure functions:

LoanStaking.lockedBalances(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1255:6:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1594:10:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 1441:14:

LoanStakingProxy.sol

Security

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 253:50:

Gas & Economy

Gas costs:

Gas requirement of function TransparentUpgradeableProxy.upgradeToAndCall is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 675:4:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 690:8:

NFTController.sol

Gas & Economy

Gas costs:

Gas requirement of function NFTController.setBoostRate is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 104:4:

Miscellaneous

Similar variable names:

NFTController.getBoostRate(address,uint256) : Variables have very similar names "token" and "tokenId". Note: Modifiers are currently not considered by this static analysis.
Pos: 82:30:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 57:8:

Solhint Linter

CarefulMath.sol

```
CarefulMath.sol:1:1: Error: Compiler version ^0.5.16 does not satisfy
the r semver requirement
```

Comptroller.sol

```
Comptroller.sol:1:1: Error: Compiler version ^0.5.16 does not satisfy
the r semver requirement
Comptroller.sol:12:26: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:87:28: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:90:28: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:93:27: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:96:26: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:249:17: Error: Avoid to make time-based decisions in
your business logic
Comptroller.sol:379:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
Comptroller.sol:593:5: Error: Explicitly mark visibility of state
Comptroller.sol:593:19: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:594:5: Error: Explicitly mark visibility of state
Comptroller.sol:594:19: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:595:5: Error: Explicitly mark visibility of state
Comptroller.sol:595:19: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:596:5: Error: Explicitly mark visibility of state
Comptroller.sol:596:19: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:1147:26: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:1171:1: Error: Contract has 18 states declarations
but allowed no more than 15
Comptroller.sol:1196:28: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:1201:28: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:1286:26: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:1294:26: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:2994:26: Error: Constant name must be in capitalized
```

```
SNAKE_CASE
Comptroller.sol:3283:5: Error: Fallback function must be simple
Comptroller.sol:3285:28: Error: Avoid using low level calls.
Comptroller.sol:3287:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
Comptroller.sol:3363:26: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:3366:29: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:3369:28: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:3372:28: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:3375:28: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:3378:28: Error: Constant name must be in capitalized
SNAKE_CASE
Comptroller.sol:3381:28: Error: Constant name must be in capitalized
SNAKE_CASE
```

ComptrollerG1.sol

```
ComptrollerG1.sol:1:1: Error: Compiler version ^0.5.16 does not
satisfy the r semver requirement
ComptrollerG1.sol:12:26: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG1.sol:87:28: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG1.sol:90:28: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG1.sol:93:27: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG1.sol:96:26: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG1.sol:249:17: Error: Avoid to make time-based decisions
in your business logic
ComptrollerG1.sol:379:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
ComptrollerG1.sol:593:5: Error: Explicitly mark visibility of state
ComptrollerG1.sol:593:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG1.sol:594:5: Error: Explicitly mark visibility of state
ComptrollerG1.sol:594:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG1.sol:595:5: Error: Explicitly mark visibility of state
ComptrollerG1.sol:595:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG1.sol:596:5: Error: Explicitly mark visibility of state
ComptrollerG1.sol:596:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG1.sol:1147:26: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG1.sol:1171:1: Error: Contract has 18 states declarations
but allowed no more than 15
ComptrollerG1.sol:1196:28: Error: Constant name must be in
```

```
capitalized SNAKE_CASE
ComptrollerG1.sol:1201:28: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG1.sol:1286:26: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG1.sol:1294:26: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG1.sol:2994:26: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG1.sol:3283:5: Error: Fallback function must be simple
ComptrollerG1.sol:3285:28: Error: Avoid using low level calls.
ComptrollerG1.sol:3287:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
ComptrollerG1.sol:3372:5: Error: Explicitly mark visibility of state
ComptrollerG1.sol:3372:19: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG1.sol:3375:5: Error: Explicitly mark visibility of state
ComptrollerG1.sol:3375:19: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG1.sol:3378:5: Error: Explicitly mark visibility of state
ComptrollerG1.sol:3378:19: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG1.sol:3381:5: Error: Explicitly mark visibility of state
ComptrollerG1.sol:3381:19: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG1.sol:3384:5: Error: Explicitly mark visibility of state
ComptrollerG1.sol:3384:19: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG1.sol:4274:17: Error: Avoid to use tx.origin
```

ComptrollerG2.sol

```
ComptrollerG2.sol:1:1: Error: Compiler version ^0.5.16 does not
satisfy the r semver requirement
ComptrollerG2.sol:12:26: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG2.sol:87:28: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG2.sol:90:28: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG2.sol:93:27: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG2.sol:96:26: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG2.sol:249:17: Error: Avoid to make time-based decisions
in your business logic
ComptrollerG2.sol:379:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
ComptrollerG2.sol:593:5: Error: Explicitly mark visibility of state
ComptrollerG2.sol:593:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG2.sol:594:5: Error: Explicitly mark visibility of state
ComptrollerG2.sol:594:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG2.sol:595:5: Error: Explicitly mark visibility of state
```

```
ComptrollerG2.sol:595:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG2.sol:596:5: Error: Explicitly mark visibility of state
ComptrollerG2.sol:596:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG2.sol:1147:26: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG2.sol:1171:1: Error: Contract has 18 states declarations
but allowed no more than 15
ComptrollerG2.sol:1196:28: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG2.sol:1201:28: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG2.sol:1286:26: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG2.sol:1294:26: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG2.sol:2994:26: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG2.sol:3283:5: Error: Fallback function must be simple
ComptrollerG2.sol:3285:28: Error: Avoid using low level calls.
ComptrollerG2.sol:3287:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
ComptrollerG2.sol:3355:28: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG2.sol:3358:28: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG2.sol:3361:28: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG2.sol:3364:28: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG2.sol:3367:28: Error: Constant name must be in
capitalized SNAKE_CASE
```

ComptrollerG3.sol

```
ComptrollerG3.sol:1:1: Error: Compiler version ^0.5.16 does not
satisfy the r semver requirement
ComptrollerG3.sol:12:26: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG3.sol:87:28: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG3.sol:90:28: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG3.sol:93:27: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG3.sol:96:26: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG3.sol:249:17: Error: Avoid to make time-based decisions
in your business logic
ComptrollerG3.sol:379:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
ComptrollerG3.sol:593:5: Error: Explicitly mark visibility of state
ComptrollerG3.sol:593:19: Error: Constant name must be in capitalized
SNAKE_CASE
```

```
ComptrollerG3.sol:594:5: Error: Explicitly mark visibility of state
ComptrollerG3.sol:594:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG3.sol:595:5: Error: Explicitly mark visibility of state
ComptrollerG3.sol:595:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG3.sol:596:5: Error: Explicitly mark visibility of state
ComptrollerG3.sol:596:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG3.sol:1147:26: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG3.sol:1171:1: Error: Contract has 18 states declarations
but allowed no more than 15
ComptrollerG3.sol:1196:28: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG3.sol:1201:28: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG3.sol:1286:26: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG3.sol:1294:26: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG3.sol:2994:26: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG3.sol:3283:5: Error: Fallback function must be simple
ComptrollerG3.sol:3285:28: Error: Avoid using low level calls.
ComptrollerG3.sol:3287:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
ComptrollerG3.sol:3353:26: Error: Constant name must be in
capitalized SNAKE_CASE
```

ComptrollerG4.sol

```
ComptrollerG4.sol:1:1: Error: Compiler version ^0.5.16 does not
satisfy the r semver requirement
ComptrollerG4.sol:12:26: Error: Constant name must be in capitalized
SNAKE_CASEComptrollerG4.sol:249:17: Error: Avoid to make time-based
decisions in your business logic
ComptrollerG4.sol:379:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
ComptrollerG4.sol:593:5: Error: Explicitly mark visibility of state
ComptrollerG4.sol:593:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG4.sol:594:5: Error: Explicitly mark visibility of state
ComptrollerG4.sol:594:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG4.sol:595:5: Error: Explicitly mark visibility of state
ComptrollerG4.sol:595:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG4.sol:596:5: Error: Explicitly mark visibility of state
ComptrollerG4.sol:596:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG4.sol:1147:26: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG4.sol:1171:1: Error: Contract has 18 states declarations
but allowed no more than 15
ComptrollerG4.sol:1196:28: Error: Constant name must be in
```

```
capitalized SNAKE_CASEComptrollerG4.sol:3283:5: Error: Fallback
function must be simple
ComptrollerG4.sol:3285:28: Error: Avoid using low level calls.
ComptrollerG4.sol:3287:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
ComptrollerG4.sol:3348:26: Error: Constant name must be in
capitalized SNAKE_CASE
```

ComptrollerG5.sol

```
ComptrollerG5.sol:1:1: Error: Compiler version ^0.5.16 does not
satisfy the r semver requirement
ComptrollerG5.sol:12:26: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG5.sol:87:28: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG5.sol:90:28: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG5.sol:93:27: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG5.sol:96:26: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG5.sol:249:17: Error: Avoid to make time-based decisions
in your business logic
ComptrollerG5.sol:379:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
ComptrollerG5.sol:593:5: Error: Explicitly mark visibility of state
ComptrollerG5.sol:593:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG5.sol:594:5: Error: Explicitly mark visibility of state
ComptrollerG5.sol:594:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG5.sol:595:5: Error: Explicitly mark visibility of state
ComptrollerG5.sol:595:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG5.sol:596:5: Error: Explicitly mark visibility of state
ComptrollerG5.sol:596:19: Error: Constant name must be in capitalized
SNAKE_CASE
ComptrollerG5.sol:1147:26: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG5.sol:1171:1: Error: Contract has 18 states declarations
but allowed no more than 15
ComptrollerG5.sol:1196:28: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG5.sol:1201:28: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerG5.sol:3283:5: Error: Fallback function must be simple
ComptrollerG5.sol:3285:28: Error: Avoid using low level calls.
ComptrollerG5.sol:3287:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
ComptrollerG5.sol:3360:26: Error: Constant name must be in
capitalized SNAKE_CASE
```

ComptrollerInterface.sol

```
ComptrollerInterface.sol:1:1: Error: Compiler version ^0.5.16 does
not satisfy the r semver requirement
ComptrollerInterface.sol:5:26: Error: Constant name must be in
capitalized SNAKE_CASE
```

ComptrollerStorage.sol

```
ComptrollerStorage.sol:1:1: Error: Compiler version ^0.5.16 does not
satisfy the r semver requirement
ComptrollerStorage.sol:212:5: Error: Explicitly mark visibility of
state
ComptrollerStorage.sol:212:19: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerStorage.sol:213:5: Error: Explicitly mark visibility of
state
ComptrollerStorage.sol:213:19: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerStorage.sol:214:5: Error: Explicitly mark visibility of
state
ComptrollerStorage.sol:214:19: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerStorage.sol:215:5: Error: Explicitly mark visibility of
state
ComptrollerStorage.sol:215:19: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerStorage.sol:765:26: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerStorage.sol:842:26: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerStorage.sol:866:1: Error: Contract has 18 states
declarations but allowed no more than 15
ComptrollerStorage.sol:891:28: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerStorage.sol:896:28: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerStorage.sol:981:26: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerStorage.sol:989:26: Error: Constant name must be in
capitalized SNAKE_CASE
ComptrollerStorage.sol:2689:26: Error: Constant name must be in
capitalized SNAKE_CASE
```

DAIInterestRateModelV3.sol

```
DAIInterestRateModelV3.sol:1:1: Error: Compiler version ^0.5.16 does
not satisfy the r semver requirement
DAIInterestRateModelV3.sol:3:1: Error: Compiler version ^0.5.16 does
not satisfy the r semver requirement
DAIInterestRateModelV3.sol:5:26: Error: Constant name must be in
capitalized SNAKE_CASE
DAIInterestRateModelV3.sol:95:26: Error: Constant name must be in
```

```
capitalized SNAKE_CASE
DAIInterestRateModelV3.sol:222:26: Error: Constant name must be in
capitalized SNAKE_CASE
DAIInterestRateModelV3.sol:224:5: Error: Explicitly mark visibility
of state
DAIInterestRateModelV3.sol:225:5: Error: Explicitly mark visibility
of state
DAIInterestRateModelV3.sol:235:161: Error: Visibility modifier must
be first in list of modifiers
DAIInterestRateModelV3.sol:249:34: Error: Variable "baseRatePerYear"
is unused
```

ErrorReporter.sol

```
ErrorReporter.sol:1:1: Error: Compiler version ^0.5.16 does not
satisfy the r semver requirement
```

Exponential.sol

```
Exponential.sol:1:1: Error: Compiler version ^0.5.16 does not satisfy
the r semver requirement
Exponential.sol:88:5: Error: Explicitly mark visibility of state
Exponential.sol:88:19: Error: Constant name must be in capitalized
SNAKE_CASE
Exponential.sol:89:5: Error: Explicitly mark visibility of state
Exponential.sol:89:19: Error: Constant name must be in capitalized
SNAKE_CASE
Exponential.sol:90:5: Error: Explicitly mark visibility of state
Exponential.sol:90:19: Error: Constant name must be in capitalized
SNAKE_CASE
Exponential.sol:91:5: Error: Explicitly mark visibility of state
Exponential.sol:91:19: Error: Constant name must be in capitalized
SNAKE_CASE
```

InterestRateModel.sol

```
InterestRateModel.sol:1:1: Error: Compiler version ^0.5.16 does not
satisfy the r semver requirement
InterestRateModel.sol:3:26: Error: Constant name must be in
capitalized SNAKE_CASE
```

JumpRateModel.sol

```
JumpRateModel.sol:1:1: Error: Compiler version ^0.5.16 does not
satisfy the r semver requirement
```

```
JumpRateModel.sol:73:26: Error: Constant name must be in capitalized
SNAKE_CASE
JumpRateModel.sol:89:26: Error: Constant name must be in capitalized
SNAKE_CASE
```

JumpRateModelV2.sol

```
JumpRateModelV2.sol:1:1: Error: Compiler version ^0.5.16 does not
satisfy the r semver requirement
JumpRateModelV2.sol:3:26: Error: Constant name must be in capitalized
SNAKE_CASE
JumpRateModelV2.sol:93:26: Error: Constant name must be in
capitalized SNAKE_CASE
```

LoanAggregatorPriceOracle.sol

```
LoanAggregatorPriceOracle.sol:1:1: Error: Compiler version ^0.5.16
does not satisfy the r semver requirement
LoanAggregatorPriceOracle.sol:334:5: Error: Explicitly mark
visibility of state
LoanAggregatorPriceOracle.sol:334:19: Error: Constant name must be in
capitalized SNAKE_CASE
LoanAggregatorPriceOracle.sol:884:26: Error: Constant name must be in
capitalized SNAKE_CASE
LoanAggregatorPriceOracle.sol:960:26: Error: Constant name must be in
capitalized SNAKE_CASE
LoanAggregatorPriceOracle.sol:983:1: Error: Contract has 18 states
declarations but allowed no more than 15
LoanAggregatorPriceOracle.sol:1008:28: Error: Constant name must be
in capitalized SNAKE_CASE
LoanAggregatorPriceOracle.sol:1013:28: Error: Constant name must be
in capitalized SNAKE_CASE
LoanAggregatorPriceOracle.sol:1098:26: Error: Constant name must be
in capitalized SNAKE_CASE
LoanAggregatorPriceOracle.sol:1106:26: Error: Constant name must be
in capitalized SNAKE_CASE
LoanAggregatorPriceOracle.sol:2941:9: Error: Avoid using inline
assembly. It is acceptable only in rare cases
```

LoanPriceOracle.sol

```
LoanPriceOracle.sol:283:5: Error: Explicitly mark visibility of state
LoanPriceOracle.sol:283:19: Error: Constant name must be in
capitalized SNAKE_CASE
LoanPriceOracle.sol:284:5: Error: Explicitly mark visibility of state
LoanPriceOracle.sol:284:19: Error: Constant name must be in
capitalized SNAKE_CASE
LoanPriceOracle.sol:834:26: Error: Constant name must be in
capitalized SNAKE_CASE
```

```
LoanPriceOracle.sol:911:26: Error: Constant name must be in
capitalized SNAKE_CASE
LoanPriceOracle.sol:934:1: Error: Contract has 18 states declarations
but allowed no more than 15
LoanPriceOracle.sol:959:28: Error: Constant name must be in
capitalized SNAKE_CASE
LoanPriceOracle.sol:964:28: Error: Constant name must be in
capitalized SNAKE_CASE
LoanPriceOracle.sol:1049:26: Error: Constant name must be in
capitalized SNAKE_CASE
LoanPriceOracle.sol:1057:26: Error: Constant name must be in
capitalized SNAKE_CASE
LoanPriceOracle.sol:2892:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
LoanPriceOracle.sol:2927:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
LoanPriceOracle.sol:2946:26: Error: Constant name must be in
capitalized SNAKE_CASE
LoanPriceOracle.sol:2976:5: Error: Explicitly mark visibility of
state
```

Maximillion.sol

```
Maximillion.sol:1:1: Error: Compiler version ^0.5.16 does not satisfy
the r semver requirement
Maximillion.sol:212:5: Error: Explicitly mark visibility of state
Maximillion.sol:212:19: Error: Constant name must be in capitalized
SNAKE_CASE
Maximillion.sol:213:5: Error: Explicitly mark visibility of state
Maximillion.sol:213:19: Error: Constant name must be in capitalized
SNAKE_CASE
Maximillion.sol:214:5: Error: Explicitly mark visibility of state
Maximillion.sol:214:19: Error: Constant name must be in capitalized
SNAKE_CASE
Maximillion.sol:215:5: Error: Explicitly mark visibility of state
Maximillion.sol:215:19: Error: Constant name must be in capitalized
SNAKE_CASE
Maximillion.sol:765:26: Error: Constant name must be in capitalized
SNAKE_CASE
Maximillion.sol:842:26: Error: Constant name must be in capitalized
SNAKE_CASE
Maximillion.sol:866:1: Error: Contract has 18 states declarations but
allowed no more than 15
Maximillion.sol:891:28: Error: Constant name must be in capitalized
SNAKE_CASE
Maximillion.sol:896:28: Error: Constant name must be in capitalized
SNAKE_CASE
Maximillion.sol:981:26: Error: Constant name must be in capitalized
SNAKE_CASE
Maximillion.sol:989:26: Error: Constant name must be in capitalized
SNAKE_CASE
Maximillion.sol:2788:5: Error: Fallback function must be simple
```

PriceOracle.sol

```
PriceOracle.sol:1:1: Error: Compiler version ^0.5.16 does not satisfy
the r semver requirement
PriceOracle.sol:212:5: Error: Explicitly mark visibility of state
PriceOracle.sol:212:19: Error: Constant name must be in capitalized
SNAKE_CASE
PriceOracle.sol:213:5: Error: Explicitly mark visibility of state
PriceOracle.sol:213:19: Error: Constant name must be in capitalized
SNAKE_CASE
PriceOracle.sol:214:5: Error: Explicitly mark visibility of state
PriceOracle.sol:214:19: Error: Constant name must be in capitalized
SNAKE_CASE
PriceOracle.sol:215:5: Error: Explicitly mark visibility of state
PriceOracle.sol:215:19: Error: Constant name must be in capitalized
SNAKE_CASE
PriceOracle.sol:765:26: Error: Constant name must be in capitalized
SNAKE_CASE
PriceOracle.sol:842:26: Error: Constant name must be in capitalized
SNAKE_CASE
PriceOracle.sol:866:1: Error: Contract has 18 states declarations but
allowed no more than 15
PriceOracle.sol:891:28: Error: Constant name must be in capitalized
SNAKE_CASE
PriceOracle.sol:2689:26: Error: Constant name must be in capitalized
SNAKE_CASE
```

Reservoir.sol

```
Reservoir.sol:1:1: Error: Compiler version ^0.5.16 does not satisfy
the r semver requirement
```

SimplePriceOracle.sol

```
SimplePriceOracle.sol:1:1: Error: Compiler version ^0.5.16 does not
satisfy the r semver requirement
SimplePriceOracle.sol:211:5: Error: Explicitly mark visibility of
state
SimplePriceOracle.sol:211:19: Error: Constant name must be in
capitalized SNAKE_CASE
SimplePriceOracle.sol:212:5: Error: Explicitly mark visibility of
state
SimplePriceOracle.sol:212:19: Error: Constant name must be in
capitalized SNAKE_CASE
SimplePriceOracle.sol:213:5: Error: Explicitly mark visibility of
state
SimplePriceOracle.sol:213:19: Error: Constant name must be in
capitalized SNAKE_CASE
```

```
SimplePriceOracle.sol:214:5: Error: Explicitly mark visibility of
state
SimplePriceOracle.sol:214:19: Error: Constant name must be in
capitalized SNAKE_CASE
SimplePriceOracle.sol:764:26: Error: Constant name must be in
capitalized SNAKE_CASE
SimplePriceOracle.sol:841:26: Error: Constant name must be in
capitalized SNAKE_CASE
SimplePriceOracle.sol:865:1: Error: Contract has 18 states
declarations but allowed no more than 15
SimplePriceOracle.sol:890:28: Error: Constant name must be in
capitalized SNAKE_CASE
SimplePriceOracle.sol:895:28: Error: Constant name must be in
capitalized SNAKE_CASE
SimplePriceOracle.sol:980:26: Error: Constant name must be in
capitalized SNAKE_CASE
SimplePriceOracle.sol:988:26: Error: Constant name must be in
capitalized SNAKE_CASE
SimplePriceOracle.sol:2688:26: Error: Constant name must be in
capitalized SNAKE_CASE
SimplePriceOracle.sol:2843:9: Error: Avoid using inline assembly. It
is acceptable only in rare cases
SimplePriceOracle.sol:2878:9: Error: Avoid using inline assembly. It
is acceptable only in rare cases
SimplePriceOracle.sol:2896:5: Error: Explicitly mark visibility of
state
```

Timelock.sol

```
Timelock.sol:1:1: Error: Compiler version ^0.5.16 does not satisfy
the r semver requirement
Timelock.sol:166:51: Error: Avoid to use ".call.value()()"
Timelock.sol:166:51: Error: Avoid using low level calls.
Timelock.sol:176:16: Error: Avoid to make time-based decisions in
your business logic
```

Unitroller.sol

```
Unitroller.sol:1:1: Error: Compiler version ^0.5.16 does not satisfy
the r semver requirement
Unitroller.sol:212:5: Error: Explicitly mark visibility of state
Unitroller.sol:212:19: Error: Constant name must be in capitalized
SNAKE_CASE
Unitroller.sol:213:5: Error: Explicitly mark visibility of state
Unitroller.sol:213:19: Error: Constant name must be in capitalized
SNAKE_CASE
Unitroller.sol:214:5: Error: Explicitly mark visibility of state
Unitroller.sol:214:19: Error: Constant name must be in capitalized
SNAKE_CASE
Unitroller.sol:215:5: Error: Explicitly mark visibility of state
Unitroller.sol:215:19: Error: Constant name must be in capitalized
SNAKE_CASE
```

```
Unitroller.sol:765:26: Error: Constant name must be in capitalized
SNAKE_CASE
Unitroller.sol:842:26: Error: Constant name must be in capitalized
SNAKE_CASE
Unitroller.sol:866:1: Error: Contract has 18 states declarations but
allowed no more than 15
Unitroller.sol:891:28: Error: Constant name must be in capitalized
SNAKE_CASE
Unitroller.sol:896:28: Error: Constant name must be in capitalized
SNAKE_CASE
Unitroller.sol:981:26: Error: Constant name must be in capitalized
SNAKE_CASE
Unitroller.sol:989:26: Error: Constant name must be in capitalized
SNAKE_CASE
Unitroller.sol:2689:26: Error: Constant name must be in capitalized
SNAKE_CASE
Unitroller.sol:2976:5: Error: Fallback function must be simple
Unitroller.sol:2978:28: Error: Avoid using low level calls.
Unitroller.sol:2980:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

WhitePaperInterestRateModel.sol

```
WhitePaperInterestRateModel.sol:1:1: Error: Compiler version ^0.5.16
does not satisfy the r semver requirement
WhitePaperInterestRateModel.sol:73:26: Error: Constant name must be
in capitalized SNAKE_CASE
WhitePaperInterestRateModel.sol:89:26: Error: Constant name must be
in capitalized SNAKE_CASE
```

YDaiDelegate.sol

```
YDaiDelegate.sol:1:1: Error: Compiler version ^0.5.16 does not
satisfy the r semver requirement
YDaiDelegate.sol:212:5: Error: Explicitly mark visibility of state
YDaiDelegate.sol:212:19: Error: Constant name must be in capitalized
SNAKE_CASE
YDaiDelegate.sol:213:5: Error: Explicitly mark visibility of state
YDaiDelegate.sol:213:19: Error: Constant name must be in capitalized
SNAKE_CASE
YDaiDelegate.sol:214:5: Error: Explicitly mark visibility of state
YDaiDelegate.sol:214:19: Error: Constant name must be in capitalized
SNAKE_CASE
YDaiDelegate.sol:215:5: Error: Explicitly mark visibility of state
YDaiDelegate.sol:215:19: Error: Constant name must be in capitalized
SNAKE_CASE
YDaiDelegate.sol:765:26: Error: Constant name must be in capitalized
SNAKE_CASE
YDaiDelegate.sol:2829:9: Error: Avoid to use inline assembly. It is
acceptable only in rare cases
YDaiDelegate.sol:2864:9: Error: Avoid to use inline assembly. It is
acceptable only in rare cases
```

```
YDaiDelegate.sol:2885:26: Error: Code contains empty blocks
YDaiDelegate.sol:3077:5: Error: Explicitly mark visibility of state
```

YErc20.sol

```
YErc20.sol:981:26: Error: Constant name must be in capitalized
SNAKE_CASE
YErc20.sol:989:26: Error: Constant name must be in capitalized
SNAKE_CASE
YErc20.sol:2829:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
YErc20.sol:2864:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

YErc20Delegate.sol

```
YErc20Delegate.sol:896:28: Error: Constant name must be in
capitalized SNAKE_CASE
YErc20Delegate.sol:981:26: Error: Constant name must be in
capitalized SNAKE_CASE
YErc20Delegate.sol:989:26: Error: Constant name must be in
capitalized SNAKE_CASE
YErc20Delegate.sol:2829:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
YErc20Delegate.sol:2864:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

YErc20Delegator.sol

```
YErc20Delegator.sol:6:26: Error: Constant name must be in capitalized
SNAKE_CASE
YErc20Delegator.sol:82:26: Error: Constant name must be in
capitalized SNAKE_CASE
YErc20Delegator.sol:106:1: Error: Contract has 18 states declarations
but allowed no more than 15
YErc20Delegator.sol:131:28: Error: Constant name must be in
capitalized SNAKE_CASE
YErc20Delegator.sol:136:28: Error: Constant name must be in
capitalized SNAKE_CASE
YErc20Delegator.sol:221:26: Error: Constant name must be in
capitalized SNAKE_CASE
YErc20Delegator.sol:229:26: Error: Constant name must be in
capitalized SNAKE_CASE
```

YErc20Immutable.sol

```
```

```
YErc20Immutable.sol:1:1: Error: Compiler version ^0.5.16 does not
satisfy the r semver requirement
YErc20Immutable.sol:215:5: Error: Explicitly mark visibility of state
YErc20Immutable.sol:215:19: Error: Constant name must be in
capitalized SNAKE_CASE
YErc20Immutable.sol:765:26: Error: Constant name must be in
capitalized SNAKE_CASE
YErc20Immutable.sol:842:26: Error: Constant name must be in
capitalized SNAKE_CASE
YErc20Immutable.sol:866:1: Error: Contract has 18 states declarations
but allowed no more than 15
YErc20Immutable.sol:891:28: Error: Constant name must be in
capitalized SNAKE_CASE
YErc20Immutable.sol:2829:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
YErc20Immutable.sol:2864:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

YEther.sol

```
YEther.sol:214:5: Error: Explicitly mark visibility of state
YEther.sol:214:19: Error: Constant name must be in capitalized
SNAKE_CASE
YEther.sol:215:5: Error: Explicitly mark visibility of state
YEther.sol:215:19: Error: Constant name must be in capitalized
SNAKE_CASE
YEther.sol:765:26: Error: Constant name must be in capitalized
SNAKE_CASE
YEther.sol:842:26: Error: Constant name must be in capitalized
SNAKE_CASE
YEther.sol:866:1: Error: Contract has 18 states declarations but
allowed no more than 15
YEther.sol:891:28: Error: Constant name must be in capitalized
SNAKE_CASE
YEther.sol:896:28: Error: Constant name must be in capitalized
SNAKE_CASE
YEther.sol:981:26: Error: Constant name must be in capitalized
SNAKE_CASE
YEther.sol:989:26: Error: Constant name must be in capitalized
SNAKE_CASE
YEther.sol:2788:5: Error: Fallback function must be simple
```

YLoanLikeDelegate.sol

```
YLoanLikeDelegate.sol:214:5: Error: Explicitly mark visibility of
state
YLoanLikeDelegate.sol:214:19: Error: Constant name must be in
capitalized SNAKE_CASE
YLoanLikeDelegate.sol:215:5: Error: Explicitly mark visibility of
state
YLoanLikeDelegate.sol:215:19: Error: Constant name must be in
capitalized SNAKE_CASE
```

```
YLoanLikeDelegate.sol:765:26: Error: Constant name must be in
capitalized SNAKE_CASE
```

YToken.sol

```
YToken.sol:214:19: Error: Constant name must be in capitalized
SNAKE_CASE
YToken.sol:215:5: Error: Explicitly mark visibility of state
YToken.sol:215:19: Error: Constant name must be in capitalized
SNAKE_CASE
YToken.sol:765:26: Error: Constant name must be in capitalized
SNAKE_CASE
YToken.sol:842:26: Error: Constant name must be in capitalized
SNAKE_CASE
YToken.sol:866:1: Error: Contract has 18 states declarations but
allowed no more than 15
YToken.sol:891:28: Error: Constant name must be in capitalized
SNAKE_CASE
YToken.sol:896:28: Error: Constant name must be in capitalized
SNAKE_CASE
YToken.sol:981:26: Error: Constant name must be in capitalized
SNAKE_CASE
YToken.sol:989:26: Error: Constant name must be in capitalized
SNAKE_CASE
```

YTokenInterfaces.sol

```
YTokenInterfaces.sol:1:1: Error: Compiler version ^0.5.16 does not
satisfy the r semver requirement
YTokenInterfaces.sol:6:26: Error: Constant name must be in
capitalized SNAKE_CASE
YTokenInterfaces.sol:82:26: Error: Constant name must be in
capitalized SNAKE_CASE
YTokenInterfaces.sol:106:1: Error: Contract has 18 states
declarations but allowed no more than 15
YTokenInterfaces.sol:131:28: Error: Constant name must be in
capitalized SNAKE_CASE
YTokenInterfaces.sol:136:28: Error: Constant name must be in
capitalized SNAKE_CASE
YTokenInterfaces.sol:221:26: Error: Constant name must be in
capitalized SNAKE_CASE
YTokenInterfaces.sol:229:26: Error: Constant name must be in
capitalized SNAKE_CASE
```

GovernorAlpha.sol

```
GovernorAlpha.sol:1:1: Error: Compiler version ^0.5.16 does not
satisfy the r semver requirement
```

```
GovernorAlpha.sol:6:28: Error: Constant name must be in capitalized
SNAKE_CASE\
GovernorAlpha.sol:241:20: Error: Avoid to make time-based decisions
in your business logic
GovernorAlpha.sol:314:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
GovernorAlpha.sol:321:5: Error: Function name must be in mixedCase
```

LOAN.sol

```
LOAN.sol:1:1: Error: Compiler version ^0.5.16 does not satisfy the r
semver requirement
LOAN.sol:15:26: Error: Constant name must be in capitalized
SNAKE_CASE
LOAN.sol:168:17: Error: Avoid to make time-based decisions in your
business logic
LOAN.sol:298:9: Error: Avoid using inline assembly. It is acceptable
only in rare cases
```

LoanLens.sol

```
LoanLens.sol:13:27: Error: Constant name must be in capitalized
SNAKE_CASE
LoanLens.sol:16:26: Error: Constant name must be in capitalized
SNAKE_CASE
LoanLens.sol:169:17: Error: Avoid to make time-based decisions in
your business logic
LoanLens.sol:299:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
LoanLens.sol:306:28: Error: Constant name must be in capitalized
SNAKE_CASE
LoanLens.sol:441:27: Error: Avoid to make time-based decisions in
your business logic
LoanLens.sol:503:20: Error: Avoid to make time-based decisions in
your business logic
LoanLens.sol:576:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
LoanLens.sol:583:5: Error: Function name must be in mixedCase
LoanLens.sol:804:5: Error: Explicitly mark visibility of state
LoanLens.sol:804:19: Error: Constant name must be in capitalized
SNAKE_CASE
LoanLens.sol:805:5: Error: Explicitly mark visibility of state
LoanLens.sol:805:19: Error: Constant name must be in capitalized
SNAKE_CASE
LoanLens.sol:806:5: Error: Explicitly mark visibility of state
LoanLens.sol:806:19: Error: Constant name must be in capitalized
SNAKE_CASE
LoanLens.sol:807:5: Error: Explicitly mark visibility of state
LoanLens.sol:807:19: Error: Constant name must be in capitalized
SNAKE_CASE
LoanLens.sol:1357:26: Error: Constant name must be in capitalized
SNAKE_CASE
```

```
LoanLens.sol:1434:26: Error: Constant name must be in capitalized
SNAKE_CASE
LoanLens.sol:1458:1: Error: Contract has 18 states declarations but
allowed no more than 15
LoanLens.sol:1483:28: Error: Constant name must be in capitalized
SNAKE_CASE
LoanLens.sol:1488:28: Error: Constant name must be in capitalized
SNAKE_CASE
LoanLens.sol:1573:26: Error: Constant name must be in capitalized
SNAKE_CASE
LoanLens.sol:1581:26: Error: Constant name must be in capitalized
SNAKE_CASE
LoanLens.sol:3421:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
LoanLens.sol:3456:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
LoanLens.sol:3476:26: Error: Constant name must be in capitalized
SNAKE_CASE
LoanLens.sol:3612:48: Error: Code contains empty blocks
```

LoanReserve.sol

```
LoanReserve.sol:351:18: Error: Parse error: missing ';' at '{'
```

LoanStaking.sol

```
LoanStaking.sol:156:18: Error: Parse error: missing ';' at '{'
LoanStaking.sol:169:18: Error: Parse error: missing ';' at '{'
LoanStaking.sol:181:18: Error: Parse error: missing ';' at '{'
LoanStaking.sol:198:18: Error: Parse error: missing ';' at '{'
LoanStaking.sol:210:18: Error: Parse error: missing ';' at '{'
LoanStaking.sol:306:18: Error: Parse error: missing ';' at '{'
LoanStaking.sol:329:18: Error: Parse error: missing ';' at '{'
LoanStaking.sol:355:18: Error: Parse error: missing ';' at '{'
LoanStaking.sol:412:18: Error: Parse error: missing ';' at '{'
LoanStaking.sol:556:18: Error: Parse error: missing ';' at '{'
LoanStaking.sol:924:18: Error: Parse error: missing ';' at '{'
```

LoanStakingProxy.sol

```
LoanStakingProxy.sol:4:1: Error: Compiler version ^0.8.4 does not
satisfy the r semver requirement
LoanStakingProxy.sol:12:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
LoanStakingProxy.sol:74:49: Error: Code contains empty blocks
LoanStakingProxy.sol:131:28: Error: Avoid using low level calls.
LoanStakingProxy.sol:203:51: Error: Avoid using low level calls.
LoanStakingProxy.sol:253:51: Error: Avoid using low level calls.
```

```
LoanStakingProxy.sol:304:13: Error: Avoid using inline assembly. It
is acceptable only in rare cases
LoanStakingProxy.sol:336:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
LoanStakingProxy.sol:346:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
LoanStakingProxy.sol:356:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
LoanStakingProxy.sol:366:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
LoanStakingProxy.sol:566:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
LoanStakingProxy.sol:603:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
LoanStakingProxy.sol:697:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
LoanStakingProxy.sol:697:114: Error: Code contains empty blocks
```

NFTController.sol

```
NFTController.sol:3:1: Error: Compiler version 0.8.4 does not satisfy
the r semver requirement
NFTController.sol:22:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
NFTController.sol:78:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
NFTController.sol:78:19: Error: Code contains empty blocks
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

Email: audit@EtherAuthority.io