Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Forest Financial
Website: forest.financial
Platform: Avalanche Network
Language: Solidity
Date: July 12th, 2022

Table of contents

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 Forest Financial to perform the Security audit of the Forest Financial 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 July 12th, 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

- Forest Financial Protocol is a DaaS (DeFi as a Service) project which has functions like receive, init, pause, unpause, burn, burnFrom, mint, etc.
- The Forest Financial contract inherits the Ownable, SafeMath, ERC721, ERC721Enumerable, Pausable, Counters, ERC20, ERC20Burnable, Context, ReentrancyGuard, Strings standard smart contracts from the OpenZeppelin library.
- These OpenZeppelin contracts are considered community-audited and time-tested, and hence are not part of the audit scope.

Audit scope

Name	**Code Review and Security Analysis Report for Forest Financial Protocol Smart Contracts**
Platform	**Avalanche / Solidity**
File 1	Diamond.sol
File 1 MD5 Hash	3CAF206EA628FB54D6314513F117B80F
File 2	DiamondInit.sol
File 2 MD5 Hash	35E9532EF83D55D88154991AC26EA3E2
File 3	ForesterNFT.sol
File 3 MD5 Hash	307F8E342D7CAE07B459A73A0CCEE4E1

File 4	ForestToken.sol
File 4 MD5 Hash	4B88D9ACAA2369C47318EC525ADCABA3
File 5	RootsToken.sol
File 5 MD5 Hash	B482D48733D34A12FB69C8E21E125633
File 6	DiamondCutFacet.sol
File 6 MD5 Hash	85257B2135726FDF21CD081C6054FAD9
File 7	DiamondLoupeFacet.sol
File 7 MD5 Hash	C2A289958965B0DF4044CB2E8599EB54
File 8	OwnershipFacet.sol
File 8 MD5 Hash	7685EDBAD7E79C1EF5311A55916ADD06
File 9	CoreNFTManageFacet.sol
File 9 MD5 Hash	26DF920E00CFD7F06355DF15A83EBAF5
File 10	HeadquarterManageFacet.sol
File 10 MD5 Hash	D875802A43BD3F3C05EB5EF5E3405CDE
File 11	ManagerFacet.sol
File 11 MD5 Hash	6946395A44F49B0CDF20E90038F0D099
File 12	ProtocolDataManageFacet.sol
File 12 MD5 Hash	C35D12327EA5CACAC2D1100D600C01E1
Updated File 12 MD5 Hash	247C1E031C72B7F53FF96A825ABB6F95
File 13	RootsManageFacet.sol
File 13 MD5 Hash	C4AE5DC44A6ED661522BA47040C34336
Updated File 13 MD5 Hash	F843922A3E97273468DB6BE5B68BB258
File 14	YieldTreeManageFacet.sol
File 14 MD5 Hash	11F4604C2C4CB4223C0925319D60B3DC
Updated File 14 MD5 Hash	C500A3325BD62DB977CFDFB5BEC0DACF
File 15	CoreNFTFacet.sol
File 15 MD5 Hash	96CC216CE9A88D64C8A36905D2A427E4

File 16	RootsMainFacet.sol
File 16 MD5 Hash	A01F77FF48C133A7924ADFE9B3E63366
Updated File 16 MD5 Hash	FCE4B78C7B2EC603E123053C98E5804D
File 17	HeadquarterMainFacet.sol
File 17 MD5 Hash	F7FDD622F09BDBA3E8A130ACE8DBB1A0
Updated File 17 MD5 Hash	E413C0DB6D3B67DAC39388A4A367A1E4
File 18	PresaleFacet.sol
File 18 MD5 Hash	E2DD4B139B331B58A0BDCB6DF5A820AB
Updated File 18 MD5 Hash	731F5A91A1CB0F851A1CA77175997456
File 19	YieldTreeClaimFacet.sol
File 19 MD5 Hash	B831F09A07A87DCD38804B15358B8EC4
Updated File 19 MD5 Hash	28C15A609756F1231ED21B38C2DC23B7
File 20	YieldTreeFeeFacet.sol
File 20 MD5 Hash	e6ade528fd36d6c1a66eeac27851d759
Updated File 20 MD5 Hash	D21C63623842D445759BB4C56069F2D1
File 21	YieldTreeMainFacet.sol
File 21 MD5 Hash	CFD5D813CC57F4FEEC130DFE6DE2BD26
Updated File 21 MD5 Hash	CF9071CACFF351854BDE4B7BC8E79517
File 22	SellTax.sol
File 22 MD5 Hash	DDF13A61F9363EE887F5D0C050B5B2C4
File 23	YieldTreeCompoundFacet.sol
File 23 MD5 Hash	AE89ED472E6B93F9BCE66156C3B4F8BB
File 24	YieldTreeGetterFacet.sol
File 24 MD5 Hash	07F7CAE9B80C30373207487B02AF61E4
Audit Date	July 12th,2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 Diamond.sol • Diamond has functions like: fallback, etc.	**YES, This is valid.**
File 2 DiamondInit.sol • DiamondInit has functions like: init.	**YES, This is valid.**
File 3 ForesterNFT.sol • Name: Forester NFT • Symbol: FORESTER • ForesterNFT owners can add a new variant and remove variant.	**YES, This is valid.**
File 4 ForestToken.sol • Name: Forest Token • Symbol: FOREST • ForestToken owners can set tax requirements.	**YES, This is valid.**
File 5 RootsToken.sol • Name: Roots Token • Symbol: ROOTS	**YES, This is valid.**
File 6 DiamondCutFacet.sol • DiamondCutFacet has functions like: diamondCut.	**YES, This is valid.**
File 7 DiamondLoupeFacet.sol • Diamond Loupe Functions like: facets, facetFunctionSelectors, etc.	**YES, This is valid.**
File 8 CoreNFTFacet.sol • CoreNFTFacet has functions like: attaching a CoreNFT to a YieldTree.	**YES, This is valid.**
File 9 HeadquarterManageFacet.sol	**YES, This is valid.**

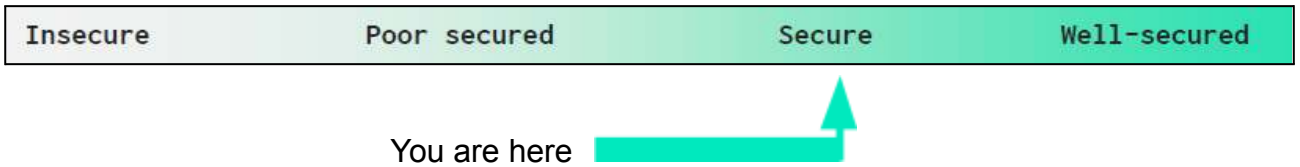
• HeadquarterManageFacet can set maximum balance and maximum level balance.	
File 10 OwnershipFacet.sol • OwnershipFacet can transfer new Ownership.	**YES, This is valid.**
File 11 PresaleFacet.sol • PresaleFacet has functions like: redeemPresale, etc. • PresaleFacet can redeem the caller of his Presale items.	**YES, This is valid.**
File 12 RootsManageFacet.sol • RootsManageFacet has functions like: setRootsGrowthFactorCap, etc.	**YES, This is valid.**
File 13 CoreNFTManageFacet.sol • CoreNFTManageFacet owner can set seed NFTBoost value. • CoreNFTManageFacet owner can set sapling NFT Boost.	**YES, This is valid.**
File 14 HeadquarterMainFacet.sol • HeadquarterMainFacet can minting a new Headquarter. • HeadquarterMainFacet can upgrade an existing Headquarter.	**YES, This is valid.**
File 15 ManagerFacet.sol • ManagerFacet owner can set gifting a YieldTree to give address.	**YES, This is valid.**
File 16 ProtocolDataManageFacet.sol • ProtocolDataManageFacet owner can set treasury address, RewardPool address, ForestToken address, etc.	**YES, This is valid.**

File 17 RootsMainFacet.sol • RootsMainFacet owner can swap Forest to Roots tokens and swapping Roots to Forest tokens.	**YES, This is valid.**
File 18 YieldTreeManageFacet.sol • YieldTreeManageFacet owner can initialize yield tree values. • YieldTreeManageFacet owner can set forest price, percentage in ether value, etc.	**YES, This is valid.**
File 19 YieldTreeClaimFacet.sol • YieldTreeClaimFacet owner can claim rewards of specific YieldTree.	**YES, This is valid.**
File 20 YieldTreeMainFacet.sol • YieldTreeMainFacet can distribute the payment of a YieldTree. • YieldTreeMainFacet can minting a YieldTree.	**YES, This is valid.**
File 21 YieldTreeFeeFacet.sol • YieldTreeFeeFacet owner can distribute the fee payment. • YieldTreeFeeFacet owner can pay fees of a YieldTree.	**YES, This is valid.**
File 22 SellTax.sol • The SellTax owner can set the tax percentage. • The SellTax owner can set the address where the tax funds will go to. • SellTax owners can update or set an address in the taxAddresses mapping. If set to true, tax has to be paid if sent to that address.	**YES, This is valid.**
File 23 YieldTreeCompoundFacet.sol • YieldTreeCompoundFacet can compound rewards into a new YieldTree.	**YES, This is valid.**

File 24 YieldTreeGetterFacet.sol ● YieldTreeGetterFacet can return the total amount of YieldTrees in existence. ● YieldTreeGetterFacet can return the total price in Forest to buy YieldTree.	**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 1 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	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	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	Moderated
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	Passed
	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**

Code Quality

This audit scope has 24 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 Forest Financial 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 Forest Financial Protocol.

The Forest Financial 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 Forest Financial Protocol smart contract code in the form of a Github weblink. 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.

Another source of information was its official website links:

- Main Website: https://forest.financial
- NFTs Website: https://nft.forest.financial
- Presale Website: https://presale.forest.financial

which provided rich information about the project architecture.

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

Diamond.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	fallback	external	Passed	No Issue
3	receive	external	Passed	No Issue

DiamondInit.sol

Functions

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

ForesterNFT.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	whenNotPaused	modifier	Passed	No Issue
3	whenPaused	modifier	Passed	No Issue
4	paused	read	Passed	No Issue
5	_requireNotPaused	internal	Passed	No Issue
6	_requirePaused	internal	Passed	No Issue
7	_pause	internal	Passed	No Issue
8	_unpause	internal	Passed	No Issue
9	onlyOwner	modifier	Passed	No Issue
10	owner	read	Passed	No Issue
11	_checkOwner	internal	Passed	No Issue
12	renounceOwnership	write	access only Owner	No Issue
13	transferOwnership	write	access only Owner	No Issue
14	_transferOwnership	internal	Passed	No Issue
15	supportsInterface	read	Passed	No Issue
16	balanceOf	read	Passed	No Issue
17	ownerOf	read	Passed	No Issue
18	name	read	Passed	No Issue
19	symbol	read	Passed	No Issue
20	tokenURI	read	Passed	No Issue
21	_baseURI	internal	Passed	No Issue
22	approve	write	Passed	No Issue

23	getApproved	read	Passed	No Issue
24	setApprovalForAll	write	Passed	No Issue
25	isApprovedForAll	read	Passed	No Issue
26	transferFrom	write	Passed	No Issue
27	safeTransferFrom	write	Passed	No Issue
28	safeTransferFrom	write	Passed	No Issue
29	_safeTransfer	internal	Passed	No Issue
30	_exists	internal	Passed	No Issue
31	_isApprovedOrOwner	internal	Passed	No Issue
32	_safeMint	internal	Passed	No Issue
33	_safeMint	internal	Passed	No Issue
34	_mint	internal	Passed	No Issue
35	_burn	internal	Passed	No Issue
36	_transfer	internal	Passed	No Issue
37	_approve	internal	Passed	No Issue
38	_setApprovalForAll	internal	Passed	No Issue
39	_requireMinted	internal	Passed	No Issue
40	_checkOnERC721Received	write	Passed	No Issue
41	_beforeTokenTransfer	internal	Passed	No Issue
42	_afterTokenTransfer	internal	Passed	No Issue
43	supportsInterface	read	Passed	No Issue
44	tokenOfOwnerByIndex	read	Passed	No Issue
45	totalSupply	read	Passed	No Issue
46	tokenByIndex	read	Passed	No Issue
47	_beforeTokenTransfer	internal	Passed	No Issue
48	_addTokenToOwnerEnumeration	write	Passed	No Issue
49	_addTokenToAllTokensEnumeration	write	Passed	No Issue
50	_removeTokenFromOwnerEnumeration	write	Passed	No Issue
51	_removeTokenFromAllTokensEnumeration	write	Passed	No Issue
52	_mint	internal	Passed	No Issue
53	tokenURI	read	Passed	No Issue
54	random	internal	Passed	No Issue
55	getURIVariant	internal	Passed	No Issue
56	addVariant	write	access only Owner	No Issue
57	removeVariant	write	access only Owner	No Issue
58	pause	write	access only Owner	No Issue
59	unpause	write	access only Owner	No Issue
60	_beforeTokenTransfer	internal	Passed	No Issue
61	supportsInterface	read	Passed	No Issue
62	safeMint	write	access only Owner	No Issue

ForestToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	owner	read	Passed	No Issue
4	_checkOwner	internal	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	name	read	Passed	No Issue
9	symbol	read	Passed	No Issue
10	tokenURI	read	Passed	No Issue
11	_baseURI	internal	Passed	No Issue
12	approve	write	Passed	No Issue
13	getApproved	read	Passed	No Issue
14	setApprovalForAll	write	Passed	No Issue
15	isApprovedForAll	read	Passed	No Issue
16	transferFrom	write	Passed	No Issue
17	safeTransferFrom	write	Passed	No Issue
18	safeTransferFrom	write	Passed	No Issue
19	_safeTransfer	internal	Passed	No Issue
20	_exists	internal	Passed	No Issue
21	_isApprovedOrOwner	internal	Passed	No Issue
22	_safeMint	internal	Passed	No Issue
23	_safeMint	internal	Passed	No Issue
24	_mint	internal	Passed	No Issue
25	_burn	internal	Passed	No Issue
26	_transfer	internal	Passed	No Issue
27	_approve	internal	Passed	No Issue
28	_setApprovalForAll	internal	Passed	No Issue
29	_requireMinted	internal	Passed	No Issue
30	_checkOnERC721Received	write	Passed	No Issue
31	_beforeTokenTransfer	internal	Passed	No Issue
32	_afterTokenTransfer	internal	Passed	No Issue
33	setTaxPercentage	external	access only Owner	No Issue
34	setTaxAddress	external	access only Owner	No Issue
35	setExcludedAddress	external	access only Owner	No Issue
36	calculateTaxAmount	read	Passed	No Issue
37	requiresTax	read	Passed	No Issue
38	burn	write	Passed	No Issue
39	burnFrom	write	Passed	No Issue
40	setTaxRequirement	external	access only Owner	No Issue
41	transferFrom	write	Passed	No Issue
42	transfer	write	Passed	No Issue

RootsToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	burn	write	Passed	No Issue
3	burnFrom	write	Passed	No Issue
4	onlyOwner	modifier	Passed	No Issue
5	owner	read	Passed	No Issue
6	_checkOwner	internal	Passed	No Issue
7	renounceOwnership	write	access only Owner	No Issue
8	transferOwnership	write	access only Owner	No Issue
9	_transferOwnership	internal	Passed	No Issue
10	name	read	Passed	No Issue
11	symbol	read	Passed	No Issue
12	tokenURI	read	Passed	No Issue
13	_baseURI	internal	Passed	No Issue
14	approve	write	Passed	No Issue
15	getApproved	read	Passed	No Issue
16	setApprovalForAll	write	Passed	No Issue
17	isApprovedForAll	read	Passed	No Issue
18	transferFrom	write	Passed	No Issue
19	safeTransferFrom	write	Passed	No Issue
20	safeTransferFrom	write	Passed	No Issue
21	_safeTransfer	internal	Passed	No Issue
22	_exists	internal	Passed	No Issue
23	_isApprovedOrOwner	internal	Passed	No Issue
24	_safeMint	internal	Passed	No Issue
25	_safeMint	internal	Passed	No Issue
26	_mint	internal	Passed	No Issue
27	_burn	internal	Passed	No Issue
28	_transfer	internal	Passed	No Issue
29	_approve	internal	Passed	No Issue
30	_setApprovalForAll	internal	Passed	No Issue
31	_requireMinted	internal	Passed	No Issue
32	_checkOnERC721Received	write	Passed	No Issue
33	_beforeTokenTransfer	internal	Passed	No Issue
34	_afterTokenTransfer	internal	Passed	No Issue
35	mint	write	access only Owner	No Issue

DiamondCutFacet.sol

Functions

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

DiamondLoupeFacet.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	facets	external	Passed	No Issue
3	facetFunctionSelectors	external	Passed	No Issue
4	facetAddresses	external	Passed	No Issue
5	facetAddress	external	Passed	No Issue
6	supportsInterface	external	Passed	No Issue

CoreNFTManageFacet.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	initCoreNFT	external	access only Owner	No Issue
4	setSeedNFTBoost	external	access only Owner	No Issue
5	setTreeNFTBoost	external	access only Owner	No Issue
6	setSaplingNFTBoost	external	access only Owner	No Issue
7	setPeltonNFTBoost	external	access only Owner	No Issue

HeadquarterManageFacet.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	initHeadquarters	external	access only Owner	No Issue
4	setHeadquartersMaxBalance	external	access only Owner	No Issue
5	setHeadquartersMaxLevel	external	access only Owner	No Issue
6	setHeadquartersMaxYieldTreesPerLevel	external	access only Owner	No Issue
7	setHeadquartersForestPrice	external	access only Owner	No Issue

OwnershipFacet.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	transferOwnership	external	Passed	No Issue
3	owner	external	Passed	No Issue

PresaleFacet.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	notBlacklisted	modifier	Passed	No Issue
3	redeemPresale	write	Passed	No Issue
4	hasRedeemedPresale	read	Passed	No Issue

RootsManageFacet.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	initRoots	external	Discount limit not set	Refer Audit Findings
4	setRootsGrowthFactorCap	external	access only Owner	No Issue
5	setRootsMaxDiscount	external	Discount limit not set	Refer Audit Findings

CoreNFTFacet.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	ownsYieldTree	modifier	Passed	No Issue
3	notBlacklisted	modifier	Passed	No Issue
4	coreNFTOwnerCheck	internal	Passed	No Issue
5	attachCoreNFT	write	access by owner of Yield Tree	No Issue
6	detachForesterNFT	write	access by owner of Yield Tree	No Issue

7	doesYieldTreeHaveCore NFTAttached	internal	Passed	No Issue
8	isCoreNFTActive	internal	Passed	No Issue
9	getCoreNFT	read	Passed	No Issue
10	getCoreNFTBoost	read	Passed	No Issue

HeadquarterMainFacet.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	nonReentrant	modifier	Passed	No Issue
3	notBlacklisted	modifier	Passed	No Issue
4	doesNotExceedMaxBalance	modifier	Passed	No Issue
5	ownsHeadquarter	modifier	Passed	No Issue
6	isUpgradeable	modifier	Passed	No Issue
7	mintHeadquarter	write	Passed	No Issue
8	upgradeHeadquarter	write	Passed	No Issue
9	getHeadquarterForestPrice	read	Passed	No Issue
10	getMaxYieldTreeCapacityOf	read	Passed	No Issue
11	getHeadquarterBalance	read	Passed	No Issue
12	getHeadquartersOf	read	Passed	No Issue
13	getHeadquarter	read	Passed	No Issue
14	getTotalHeadquarters	read	Passed	No Issue

ManagerFacet.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	hasSpaceForYieldTree	modifier	Passed	No Issue
4	giftYieldTree	external	access only Owner	No Issue
5	addAdmin	external	access only Owner	No Issue
6	removeAdmin	external	access only Owner	No Issue
7	onlyOwnerOrAdmin	modifier	Passed	No Issue

ProtocolDataManageFacet.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	initProtocolMetaData	external	access only Owner	No Issue

4	setTreasury	external	access only Owner	No Issue
5	setRewardPool	external	access only Owner	No Issue
6	setForestToken	external	access only Owner	No Issue
7	setRootsToken	external	access only Owner	No Issue
8	setStableToken	external	access only Owner	No Issue
9	setForesterNFT	external	access only Owner	No Issue
10	setJoeRouter	external	access only Owner	No Issue
11	setJoeFactory	external	access only Owner	No Issue
12	setJoePair	external	access only Owner	No Issue
13	setPriceFeed	external	access only Owner	No Issue
14	initProcotolMetaData2	external	access only Owner	No Issue

RootsMainFacet.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	notBlacklisted	modifier	Passed	No Issue
3	swapForestToRoots	write	Passed	No Issue
4	swapRootsToForest	write	Passed	No Issue
5	getRootsBackingPrice	read	Passed	No Issue
6	getRootsBuyPrice	read	Passed	No Issue
7	getRootsSellPrice	read	Passed	No Issue
8	getRootsTreasuryBalance	read	Passed	No Issue
9	getRootsSupply	read	Passed	No Issue

YieldTreeManageFacet.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	initYieldTrees	external	access only Owner	No Issue
4	setYieldTreesForestPrice	external	Fee and Percentage limit not set	Refer Audit Findings
5	setYieldTreesPercentageIn Ether	external	Fee and Percentage limit not set	Refer Audit Findings
6	setYieldTreesBaseRewards	external	access only Owner	No Issue
7	setYieldTreesDecayAfter	external	access only Owner	No Issue
8	setYieldTreesDecayTo	external	access only Owner	No Issue
9	setYieldTreesDecayPerDay	external	access only Owner	No Issue
10	setYieldTreesForestLiquidity Percentage	external	Fee and Percentage limit not set	Refer Audit Findings
11	setYieldTreesForestReward PoolPercentage	external	Fee and Percentage limit not set	Refer Audit Findings

12	setYieldTreesForestTreasuryPercentage	external	Fee and Percentage limit not set	Refer Audit Findings
13	setYieldTreesEtherLiquidityPercentage	external	Fee and Percentage limit not set	Refer Audit Findings
14	setYieldTreesEtherRewardPercentage	external	Fee and Percentage limit not set	Refer Audit Findings
15	setYieldTreesEtherTreasuryPercentage	external	Fee and Percentage limit not set	Refer Audit Findings
16	setYieldTreesForestFeePerMonth	external	Fee and Percentage limit not set	Refer Audit Findings
17	setYieldTreesForestFeePerMonth	external	Fee and Percentage limit not set	Refer Audit Findings
18	initYieldTreesPaymentDistribution	external	access only Owner	No Issue

YieldTreeClaimFacet.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	notBlacklisted	modifier	Passed	No Issue
3	ownsYieldTree	modifier	Passed	No Issue
4	hasSpaceForYieldTree	modifier	Passed	No Issue
5	claimRewardsOfYieldTree	write	Passed	No Issue
6	claimRewardsOfAllYieldTrees	write	Passed	No Issue
7	getYieldTreeRewards	read	Passed	No Issue
8	getTotalYieldTreeRewards	read	Passed	No Issue

YieldTreeMainFacet.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	notBlacklisted	modifier	Passed	No Issue
3	ownsHeadquarter	modifier	Passed	No Issue
4	ownsYieldTree	modifier	Passed	No Issue
5	distributeYieldTreePayment	internal	Passed	No Issue
6	mintYieldTree	write	access by owns Head quarter	No Issue
7	getTotalYieldTrees	read	Passed	No Issue
8	getYieldTreeForestPrice	read	Passed	No Issue
9	getYieldTreeEtherPrice	read	Passed	No Issue
10	getYieldTreeBalance	read	Passed	No Issue
11	getYieldTreesOf	read	Passed	No Issue

12	getYieldTree	read	Passed	No Issue
13	headquarterNotOnMaxCapacity	modifier	Passed	No Issue

YieldTreeFeeFacet.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	notBlacklisted	modifier	Passed	No Issue
3	ownsYieldTree	modifier	Passed	No Issue
4	payYieldTreeFee	write	Passed	No Issue
5	payAllYieldTreeFees	write	Passed	No Issue
6	getRemainingHoursUntilFeeExpiry	read	Passed	No Issue
7	distributeYieldTreeFeePayment	internal	Passed	No Issue

SellTax.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	owner	read	Passed	No Issue
4	_checkOwner	internal	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	setTaxPercentage	external	access only Owner	No Issue
9	setTaxAddress	external	access only Owner	No Issue
10	setExcludedAddress	external	access only Owner	No Issue
11	calculateTaxAmount	read	Passed	No Issue
12	requiresTax	read	Passed	No Issue
13	setTaxReceiver	external	access only Owner	No Issue

YieldTreeCompoundFacet.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	notBlacklisted	modifier	Passed	No Issue
3	hasSpaceForYieldTree	modifier	Passed	No Issue

4	distributeYieldTreeCompoundPayment	internal	Passed	No Issue
5	compoundRewardsIntoYieldTree	write	Passed	No Issue

YieldTreeGetterFacet.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getTotalYieldTrees	read	Passed	No Issue
3	getYieldTreeFullForestPrice	read	Passed	No Issue
4	getYieldTreeForestPrice	read	Passed	No Issue
5	getYieldTreeEtherPrice	read	Passed	No Issue
6	getYieldTreeBalance	read	Passed	No Issue
7	getYieldTreesOf	read	Passed	No Issue
8	getYieldTree	read	Passed	No Issue
9	getYieldTreeRewards	read	Passed	No Issue
10	getTotalYieldTreeRewards	read	Passed	No Issue
11	getRemainingHoursUntilFeeExpiry	read	Passed	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 lose
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) Critical operation lacks event log:

Missing event log for:

YieldTreeClaimFacet.sol

- claimRewardsOfYieldTree
- claimRewardsOfAllYieldTrees

YieldTreeFeeFacet.sol

- payAllYieldTreeFees
- payYieldTreeFee

Resolution: Write an event log for listed events.

Status: Fixed

Very Low / Informational / Best practices:

(1) Discount limit not set:**- RootsManageFacet.sol**

In initRoots, setRootsMaxDiscount functions, the Owner can set the individual discount to any variable. This might deter investors as they could be wary that the discount might one day be set to any max number to force transfers to go to the contract owner.

Resolution: Consider adding an explicit limit to the maxdiscount adjustment function.

Status: **Fixed**

(2) Fee and Percentage limit not set:**- YieldTreeManageFacet.sol**

In the below functions the Owner can set the individual discount to any variable. This might deter investors as they could be wary that the discount might one day be set to any max number to force transfers to go to the contract owner:

- setYieldTreesForestPrice
- setYieldTreesForestFeePerMonth
- setYieldTreesPercentageInEther
- setYieldTreesForestLiquidityPercentage
- setYieldTreesForestRewardPoolPercentage
- setYieldTreesForestTreasuryPercentage
- setYieldTreesEtherLiquidityPercentage
- setYieldTreesEtherRewardPercentage
- setYieldTreesEtherTreasuryPercentage
- setYieldTreesFeeTreasuryPercentage
- setYieldTreesFeeCharityPercentage

Resolution: Consider adding an explicit limit to the maxdiscount adjustment function.

Status: Acknowledged.

(3) Same Contract Names:**- YieldTreeManageFacet.sol**

YieldTreeManageFacet.sol and ProtocolDataManageFacet.sol files have the same contract names.

Resolution: We suggest changing the contract names.

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:

- claimRewardsOfYieldTree: YieldTreeClaimFacet owner can claim rewards of yield tree.
- mintYieldTree: YieldTreeMainFacet owner can mint yield tree token.
- payYieldTreeFee: YieldTreeFeeFacet owner can pay yield tree fees.
- setTaxPercentage: SellTax owner can set the tax percentage.
- setTaxReceiver: SellTax owner can set receiver tax.
- setTaxAddress: SellTax owner can update or set an address in the taxAddresses mapping.
- setExcludedAddress: SellTax owner can update or set an excluded address.
- setTaxReceiver: SellTax owner can set the address where the tax funds will go to.
- addVariant: The ForesterNFT owner can add a new variant value.
- removeVariant: ForesterNFT owner can remove variant value.
- safeMint: ForesterNFT owner can safely mint a token.
- pause: ForesterNFT owner can trigger stopped state.
- unpause: ForesterNFT owner can return to normal state.
- setTaxRequirement: ForestToken owner can set tax requirement status.
- mint: RootsToken owner can mint a token from the address.
- attachCoreNFT: CoreNFTFacet owner can attach a CoreNFT to a YieldTree.
- detachForesterNFT: The ForesterNFTFacet owner can detach a Forester NFT from a YieldTree.
- upgradeHeadquarter: HeadquarterMainFacet owner can upgrade headquarter value.
- initCoreNFT: CoreNFTManageFacet owner can initialize core NFT.
- setSeedNFTBoost: CoreNFTManageFacet owner can set seed NFT Boost value.
- setSaplingNFTBoost: CoreNFTManageFacet owner can set sapling NFT Boost value.
- setTreeNFTBoost: CoreNFTManageFacet owner can set tree NFT Boost value.

- setPeltonNFTBoost: CoreNFTManageFacet owner can set pelton NFT boost value.
- initHeadquarters: HeadquarterManageFacet owner can initialize headquarters.
- setHeadquartersMaxBalance: HeadquarterManageFacet owner can set headquarters maximum balance.
- setHeadquartersMaxLevel: HeadquarterManageFacet owner can set headquarters maximum level value.
- setHeadquartersMaxYieldTreesPerLevel: HeadquarterManageFacet owner can set headquarters maximum yield trees per level.
- setHeadquartersForestPrice: HeadquarterManageFacet owner can set headquarters forest price.
- giftYieldTree: ManagerFacet owner can gift yield tree address.
- removeAdmin: ManagerFacet owner can remove an admin address to the protocol.
- addAdmin: ManagerFacet owner can add an admin address to the protocol.
- initProtocolMetaData: ProtocolDataManageFacet owner can initialize protocol metadata addresses.
- initProcotolMetaData2: ProtocolDataManageFacet owner can initialize protocol metadata addresses.
- setTreasury: ProtocolDataManageFacet owner can set a new treasury address.
- setRewardPool: ProtocolDataManageFacet owner can set a new reward pool address.
- setForestToken: ProtocolDataManageFacet owner can set a new forest token address.
- setRootsToken: ProtocolDataManageFacet owner can set a new root token address.
- setStableToken: ProtocolDataManageFacet owner can set a new stable token address.
- setForesterNFT: ProtocolDataManageFacet owner can set a new forest NFT address.
- setJoeRouter: ProtocolDataManageFacet owner can set a new Joe router address.
- setJoeFactory: ProtocolDataManageFacet owner can set a new Joe factory address.
- setJoePair: ProtocolDataManageFacet owner can set a new Joe pair address.
- setPriceFeed: ProtocolDataManageFacet owner can set a new price feed address.

- initRoots: RootsManageFacet owner can initialize root addresses.
- setRootsTreasury: RootsManageFacet owner can set root treasury address.
- setRootsAdditionalGrowthFactorCap: RootsManageFacet owner can set roots additional growth factor cap value.
- setRootsResetPeriodAfter: RootsManageFacet owner can set root reset period value.
- setRootsMaxDiscount: RootsManageFacet owner can set root maximum discount value.
- recoverRootsSupply: RootsManageFacet owner can recover root supply.
- initYieldTrees: YieldTreeManageFacet owner can initialize yield tree values.
- initYieldTreesPaymentDistribution: YieldTreeManageFacet owner can initialize yield tree payment distribution value.
- setYieldTreesForestPrice: YieldTreeManageFacet owner can set yield tree forest price.
- setYieldTreesPercentageInEther: YieldTreeManageFacet owner can set yield tree percentage in ether.
- setYieldTreesBaseRewards: YieldTreeManageFacet owner can set yield tree base rewards value.
- setYieldTreesDecayAfter: YieldTreeManageFacet owner can set after yield tree decay value
- setYieldTreesDecayTo: YieldTreeManageFacet owner can set yield tree decay.
- setYieldTreesDecayPerDay: YieldTreeManageFacet owner can set yield tree decay per day value.
- setYieldTreesForestFeePerMonth: YieldTreeManageFacet owner can set yield tree forest fee per month.
- setYieldTreesForestLiquidityPercentage: YieldTreeManageFacet owner can set yield tree forest liquidity percentage value.
- setYieldTreesForestRewardPoolPercentage: YieldTreeManageFacet owner can set yield tree forest rewards pool percentage.
- setYieldTreesForestTreasuryPercentage: YieldTreeManageFacet owner can set yield tree forest treasury percentage.
- setYieldTreesEtherLiquidityPercentage: YieldTreeManageFacet owner can set yield tree ether liquidity percentage.

- setYieldTreesEtherRewardPercentage: YieldTreeManageFacet owner can set yield tree ether reward percentage.
- setYieldTreesEtherTreasuryPercentage: YieldTreeManageFacet owner can set yield tree ether treasury percentage.
- setYieldTreesFeeTreasuryPercentage: YieldTreeManageFacet owner can set yield tree fee treasury percentage.
- setYieldTreesFeeCharityPercentage: YieldTreeManageFacet owner can set yield tree fee charity percentage.

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 Github weblink. 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"**.

Email: audit@EtherAuthority.io

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.

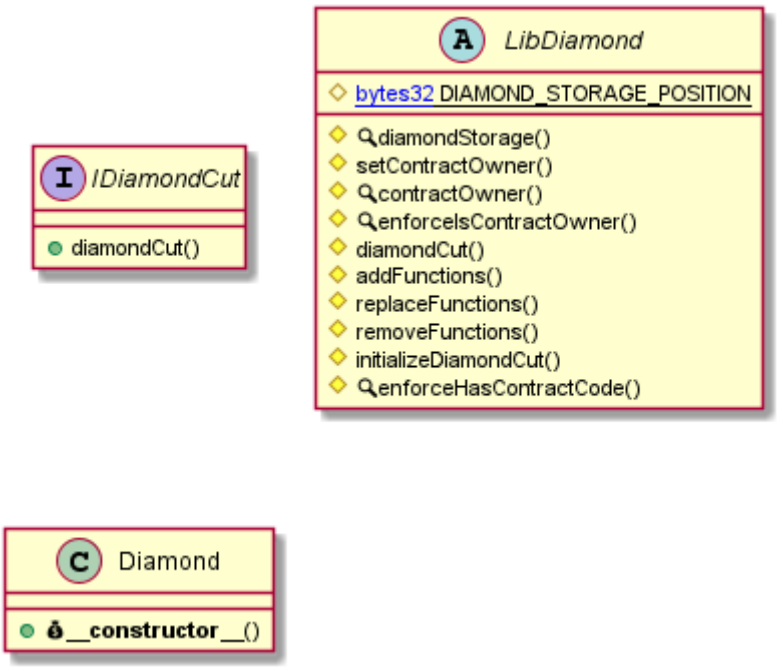
Technical Disclaimer

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.

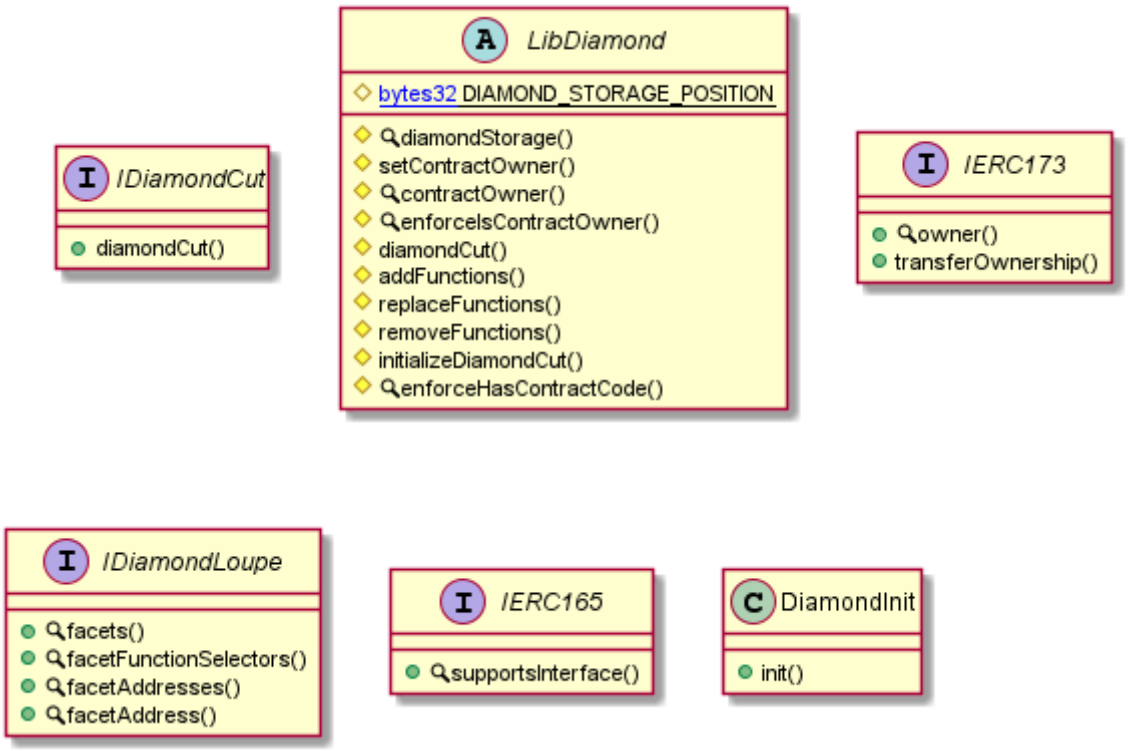
Appendix

Code Flow Diagram - Forest Financial

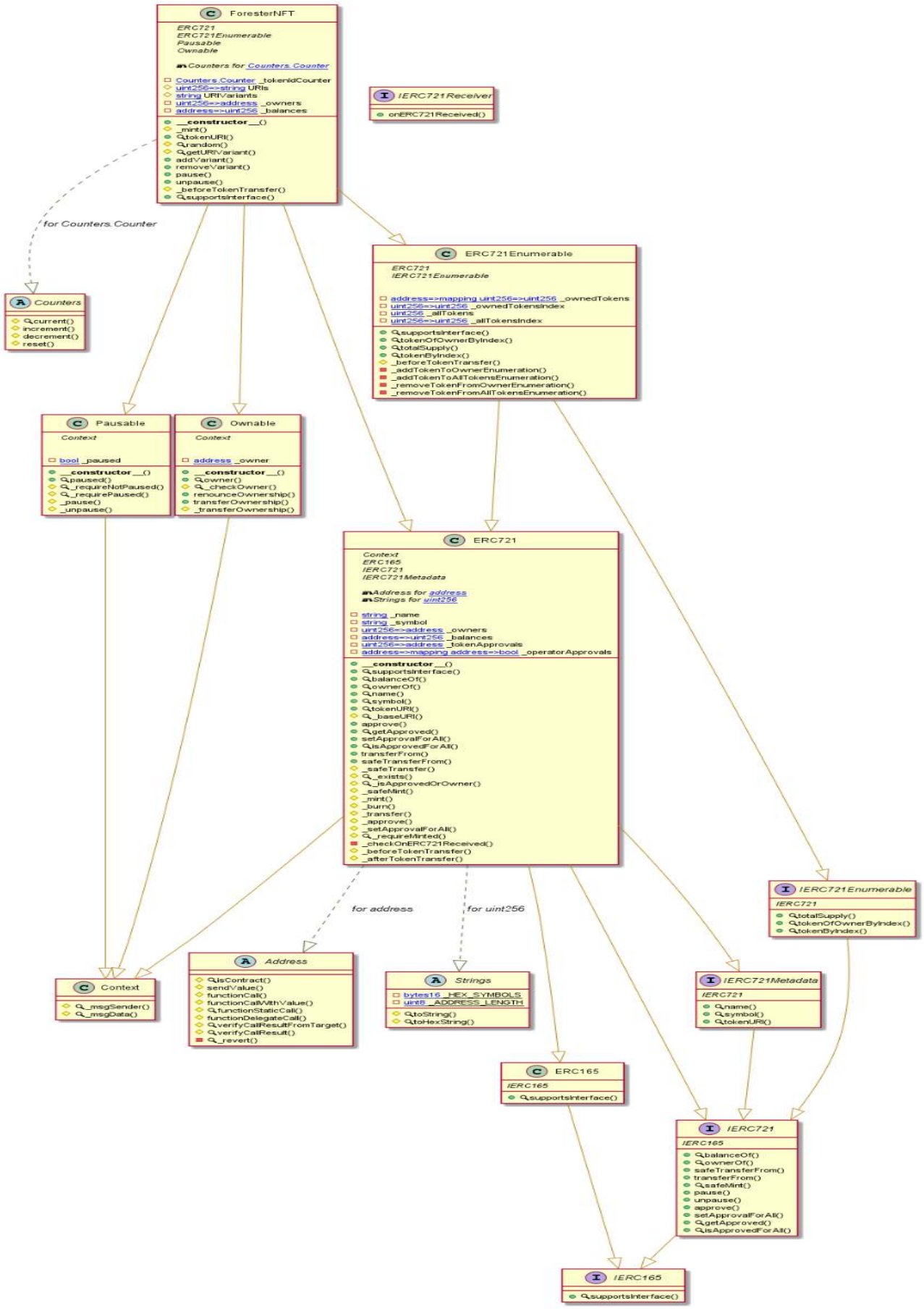
Diamond Diagram



DiamondInit Diagram

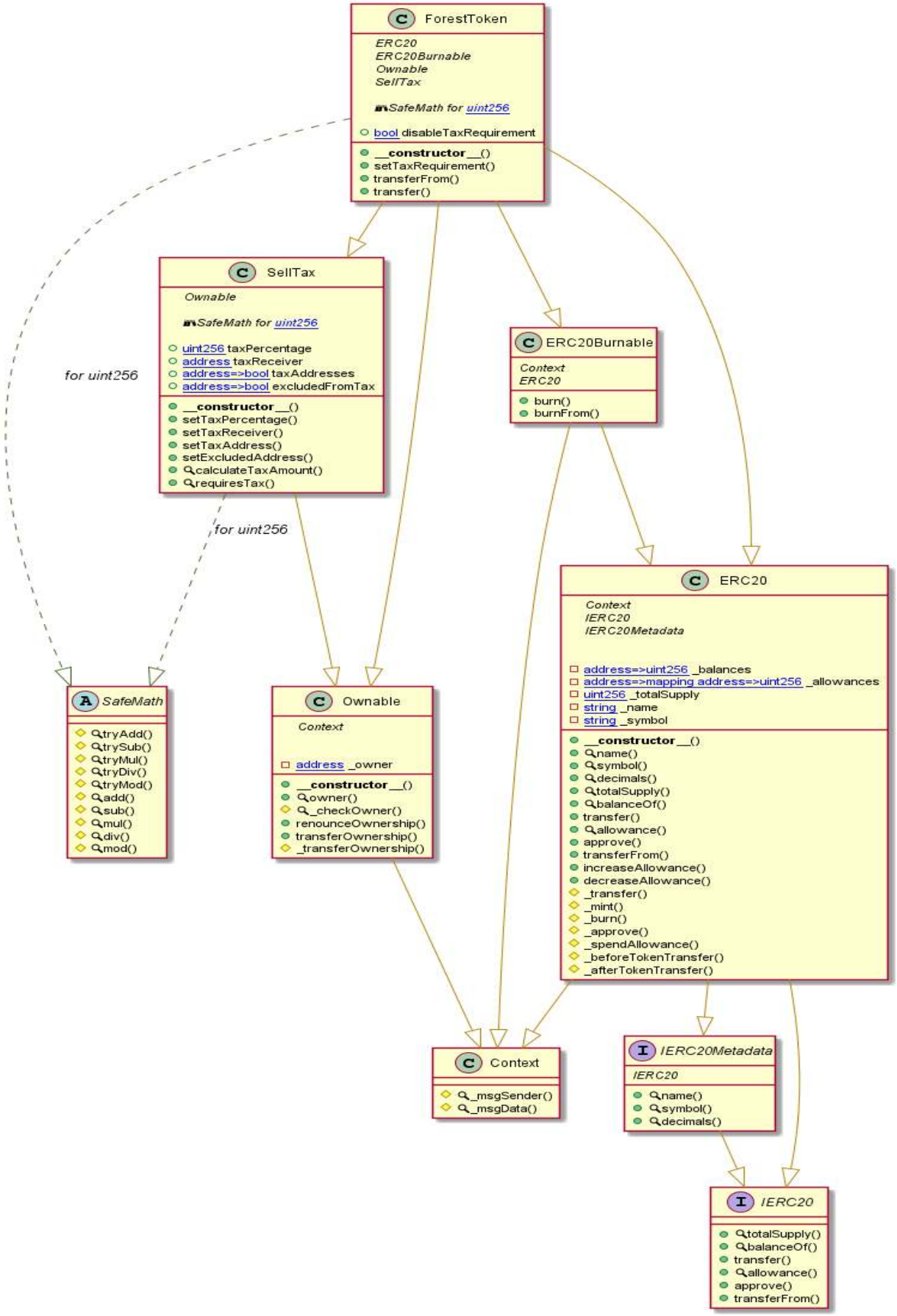


ForesterNFT Diagram

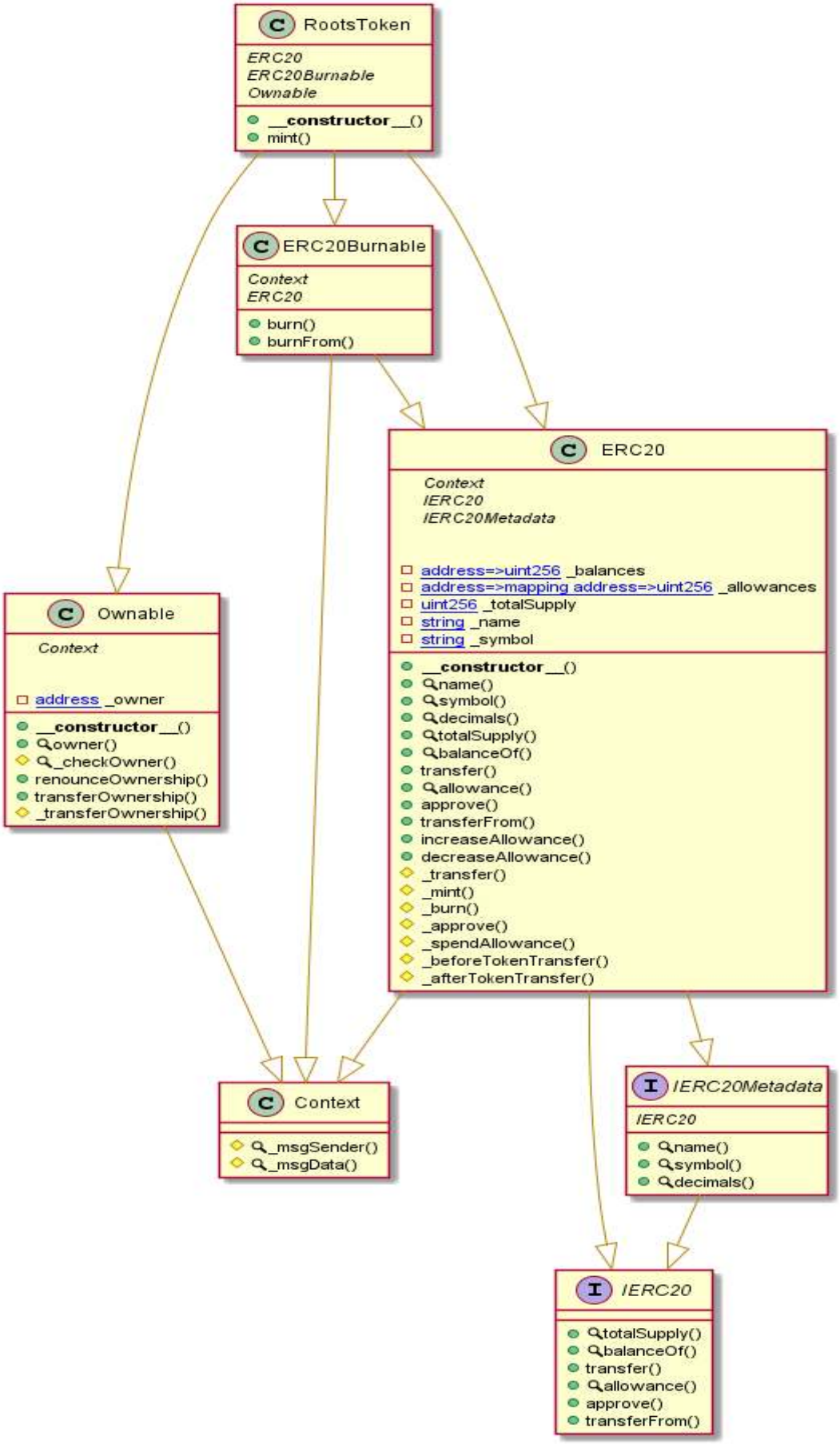


Email: audit@EtherAuthority.io

ForestToken Diagram

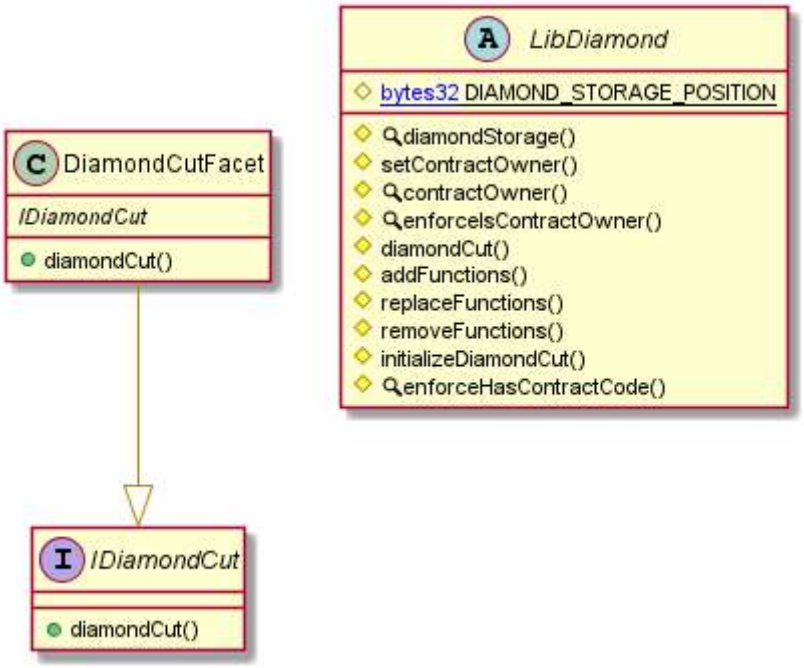


RootsToken Diagram

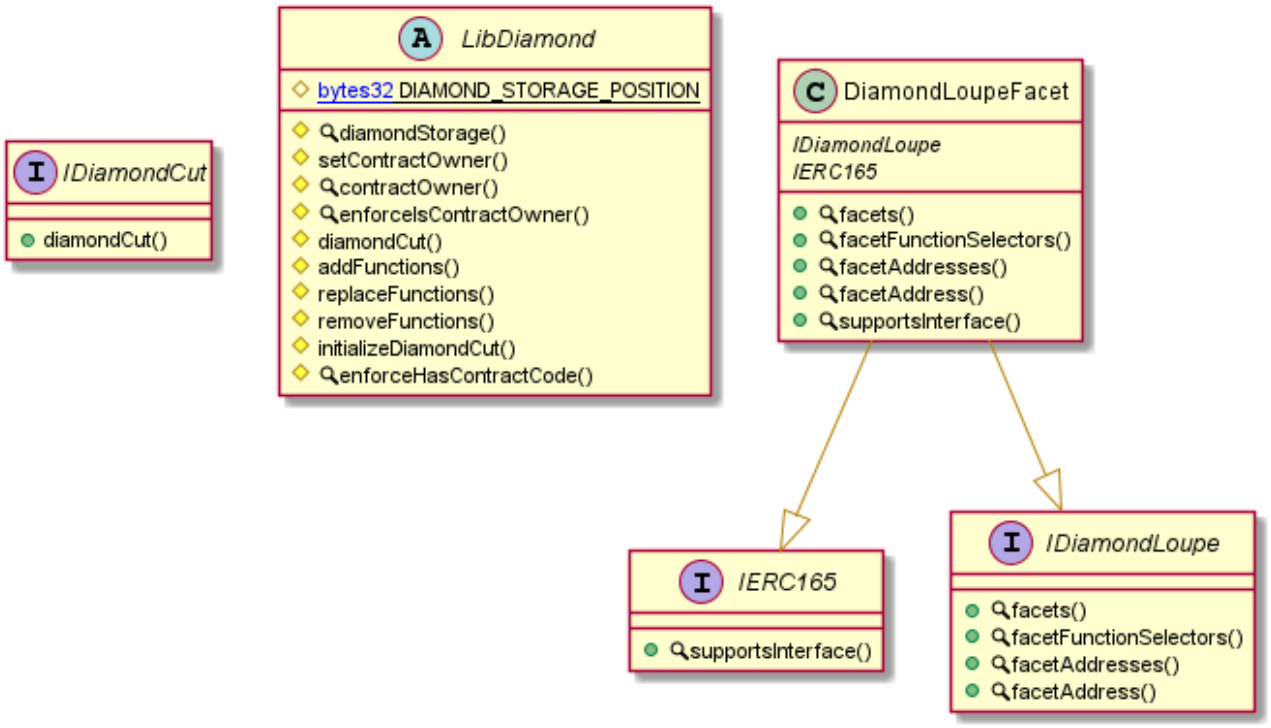


Email: audit@EtherAuthority.io

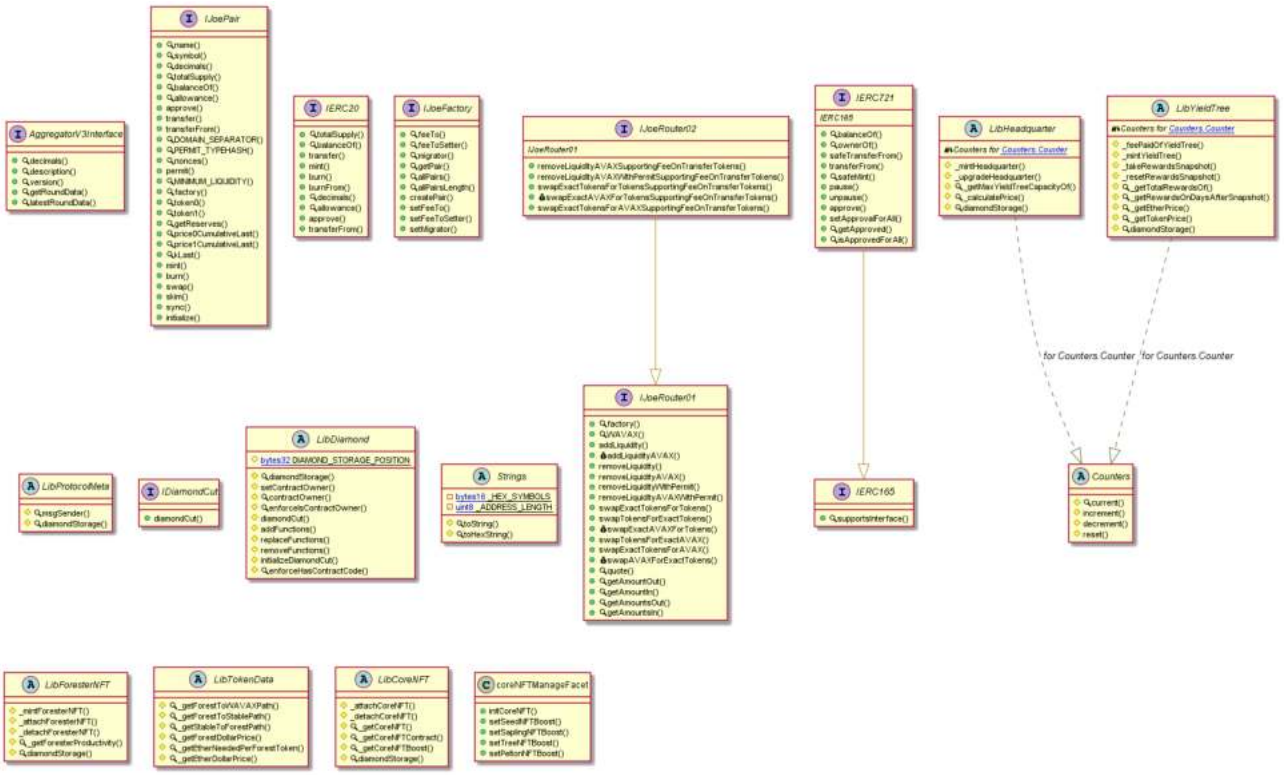
DiamondCutFacet Diagram



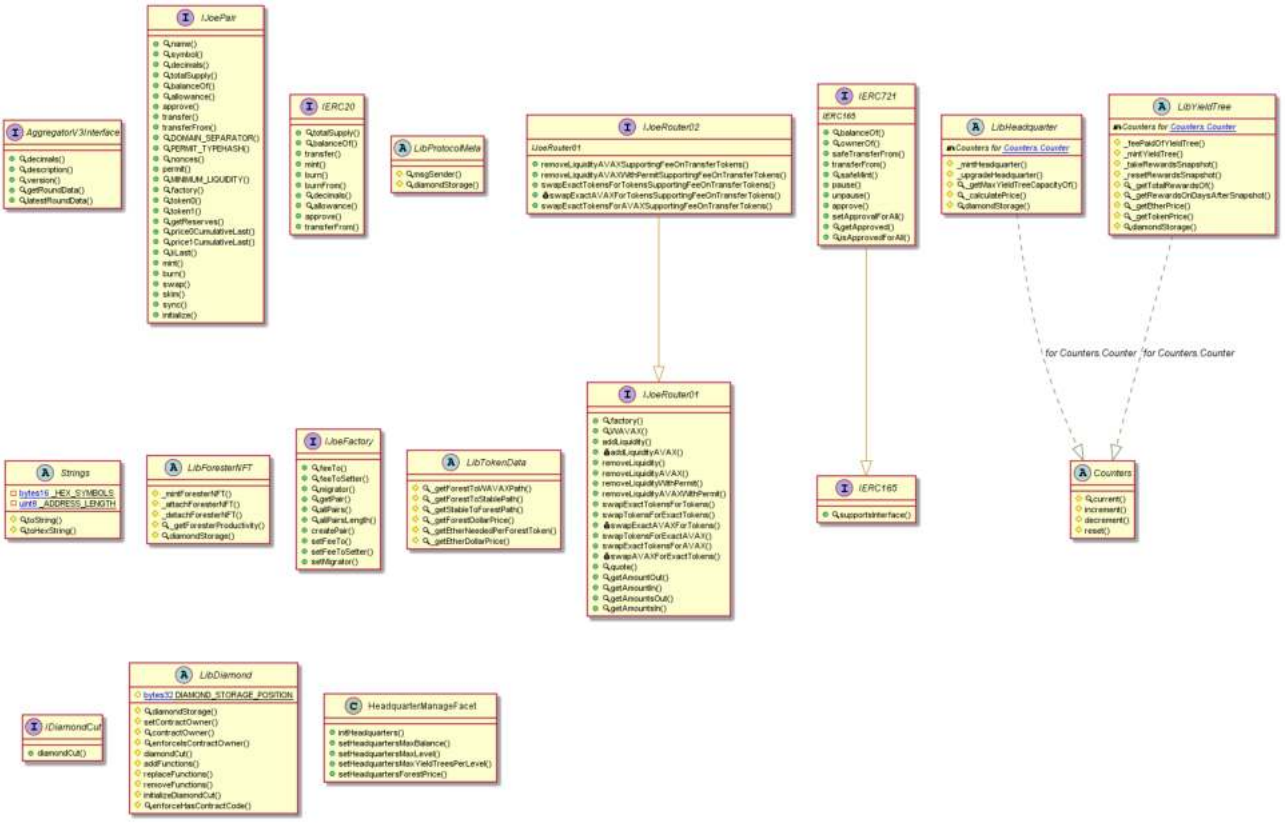
DiamondLoupeFacet Diagram



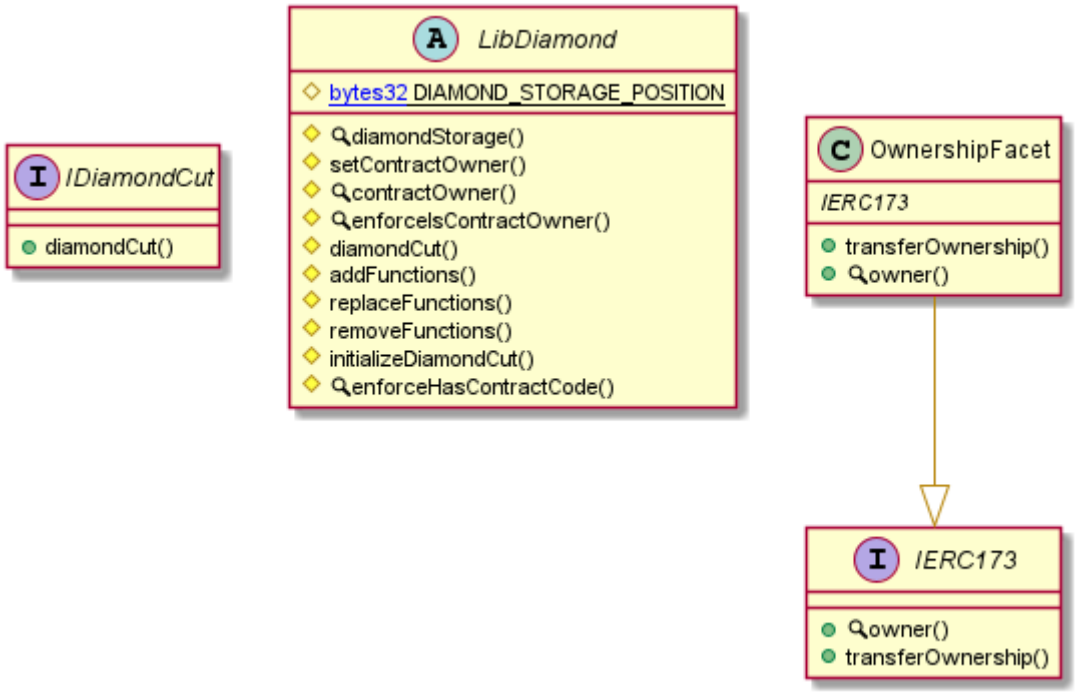
CoreNFTManageFacet Diagram



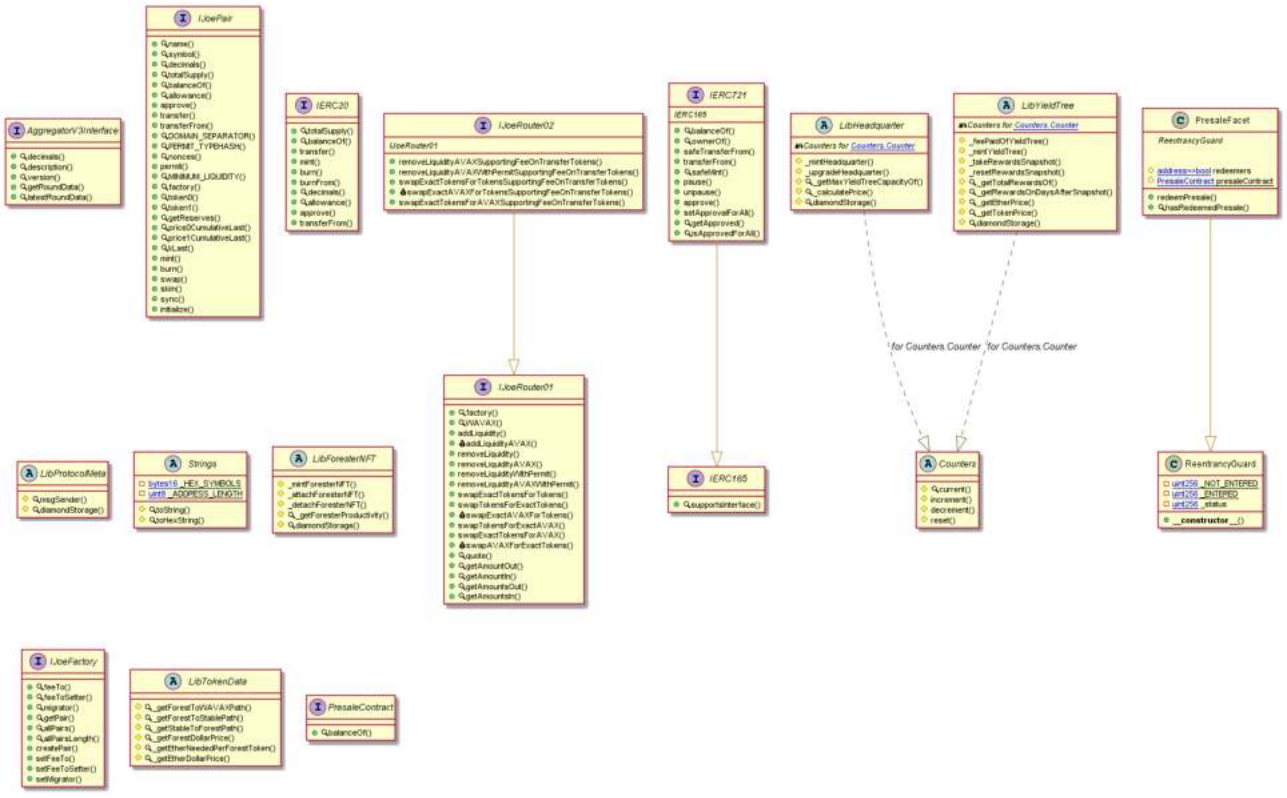
HeadquarterManageFacet Diagram



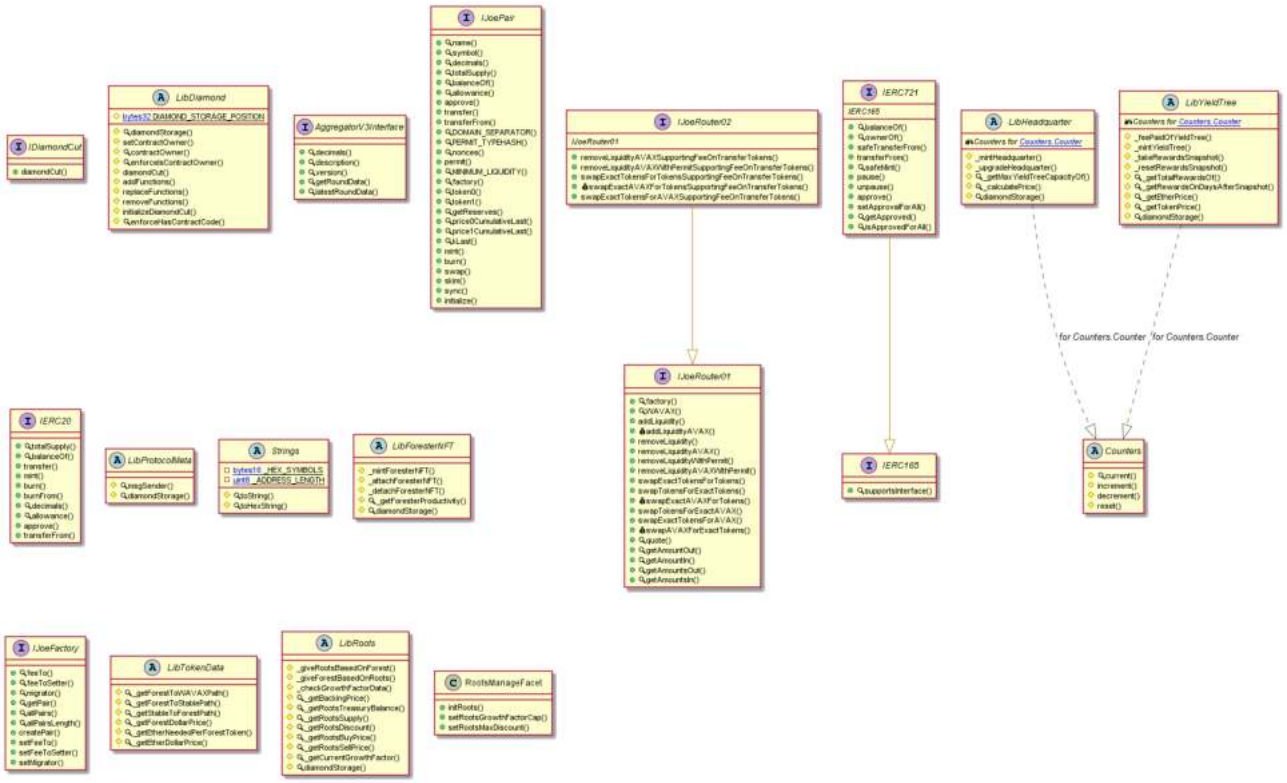
OwnershipFacet Diagram



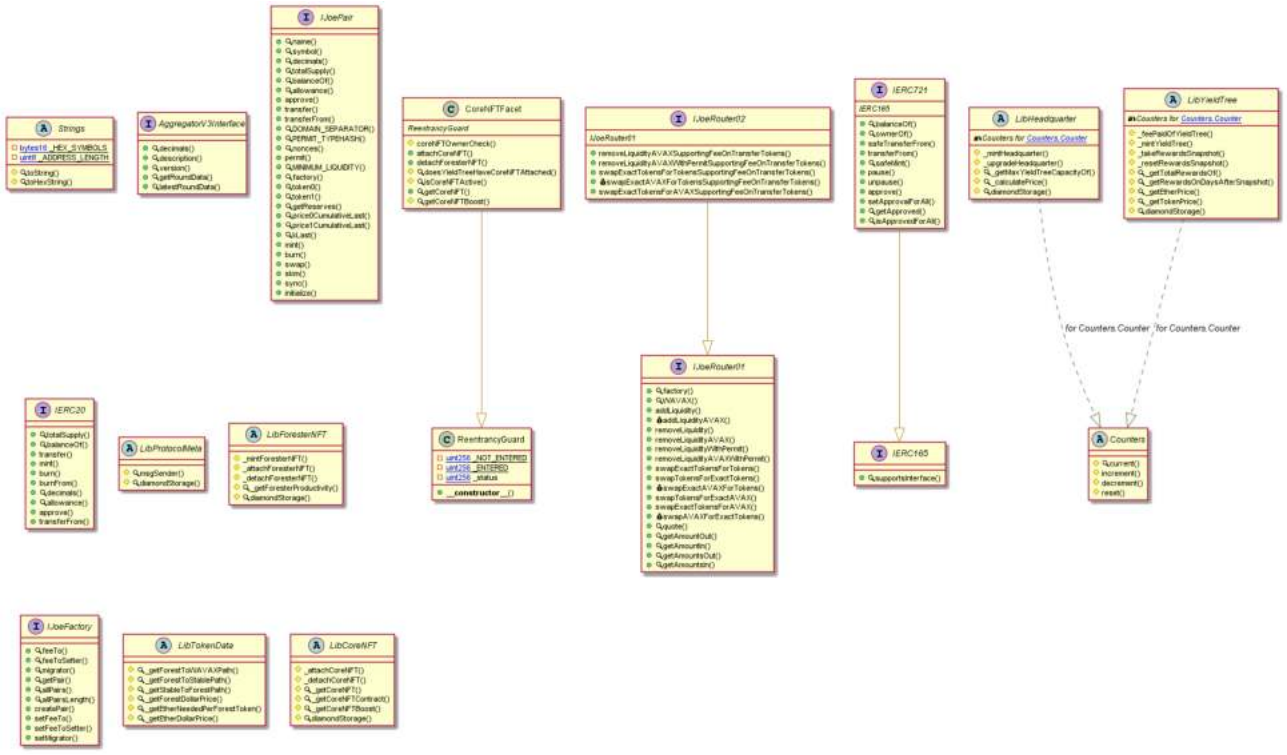
PresaleFacet Diagram



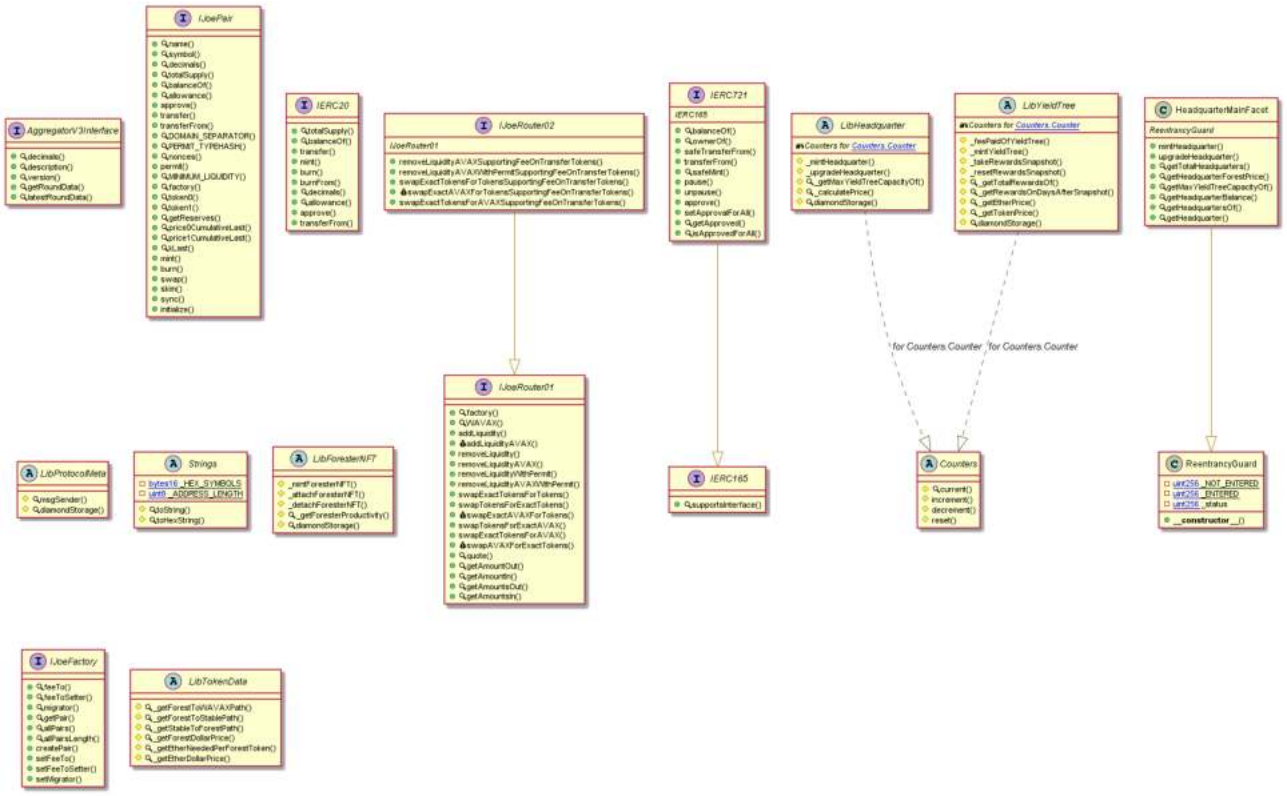
RootsManageFacet Diagram



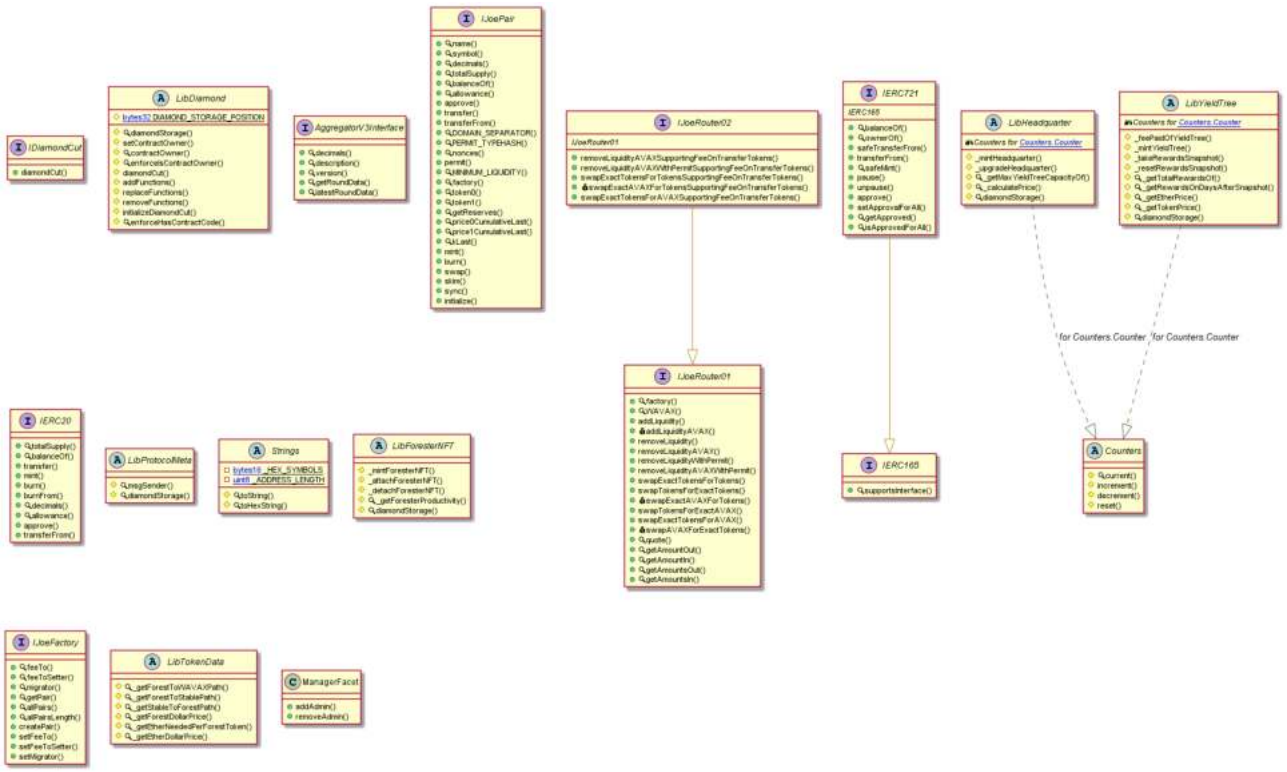
CoreNFTFacet Diagram



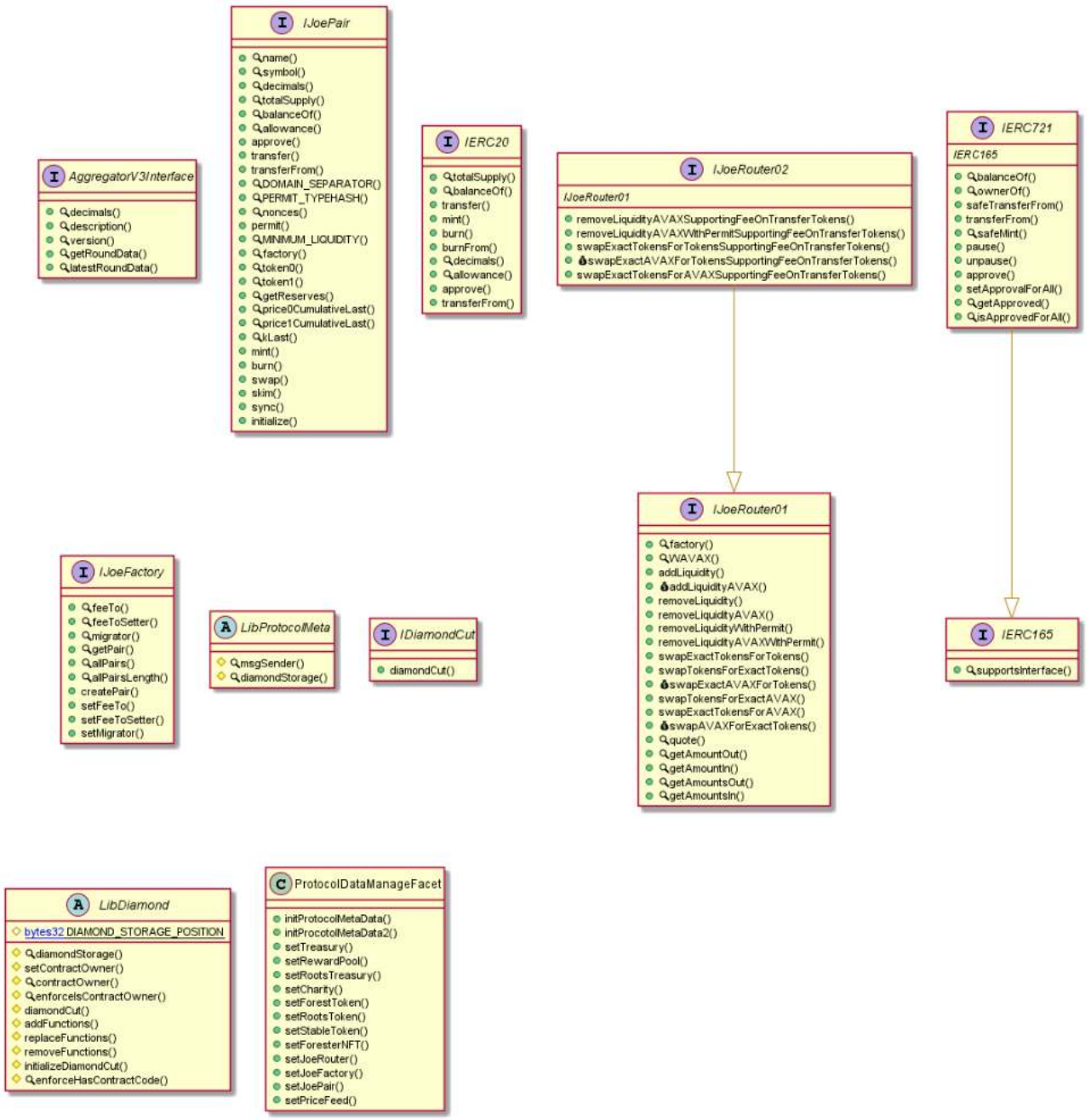
HeadquarterMainFacet Diagram



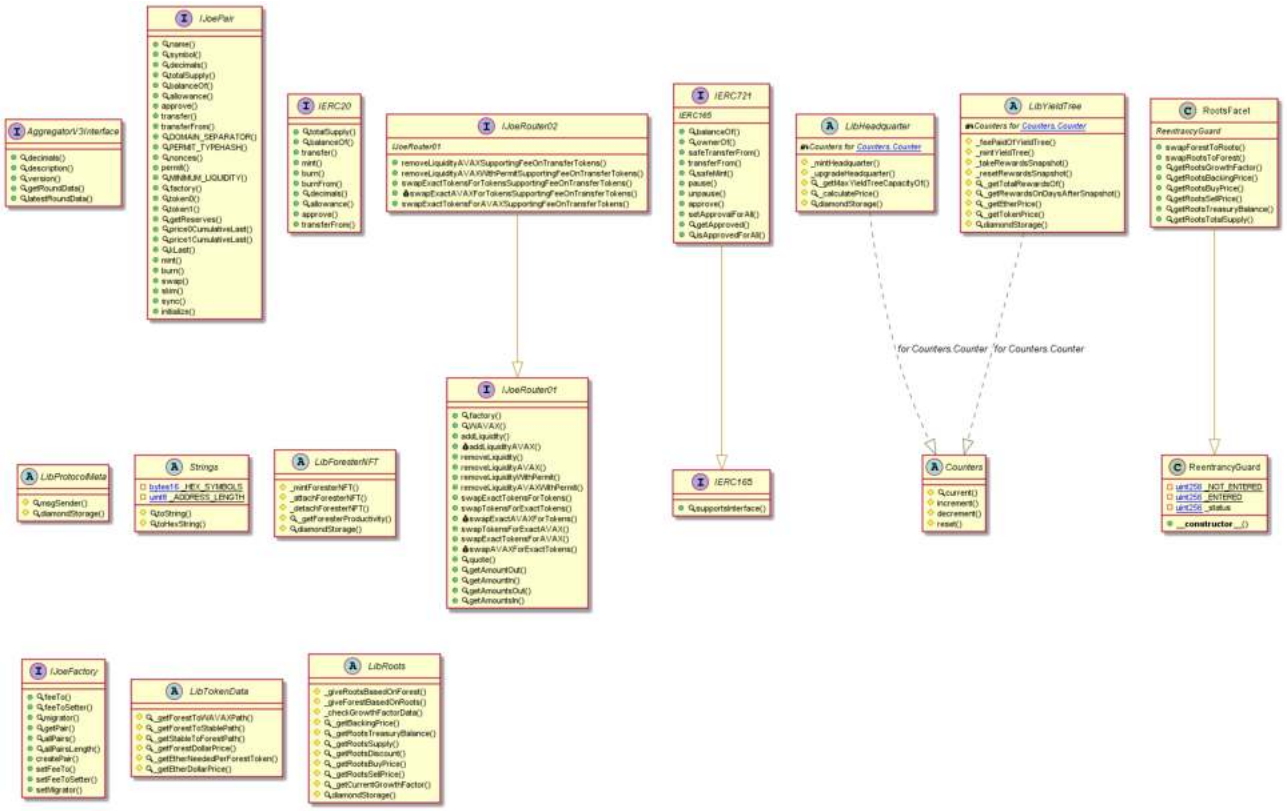
ManagerFacet Diagram



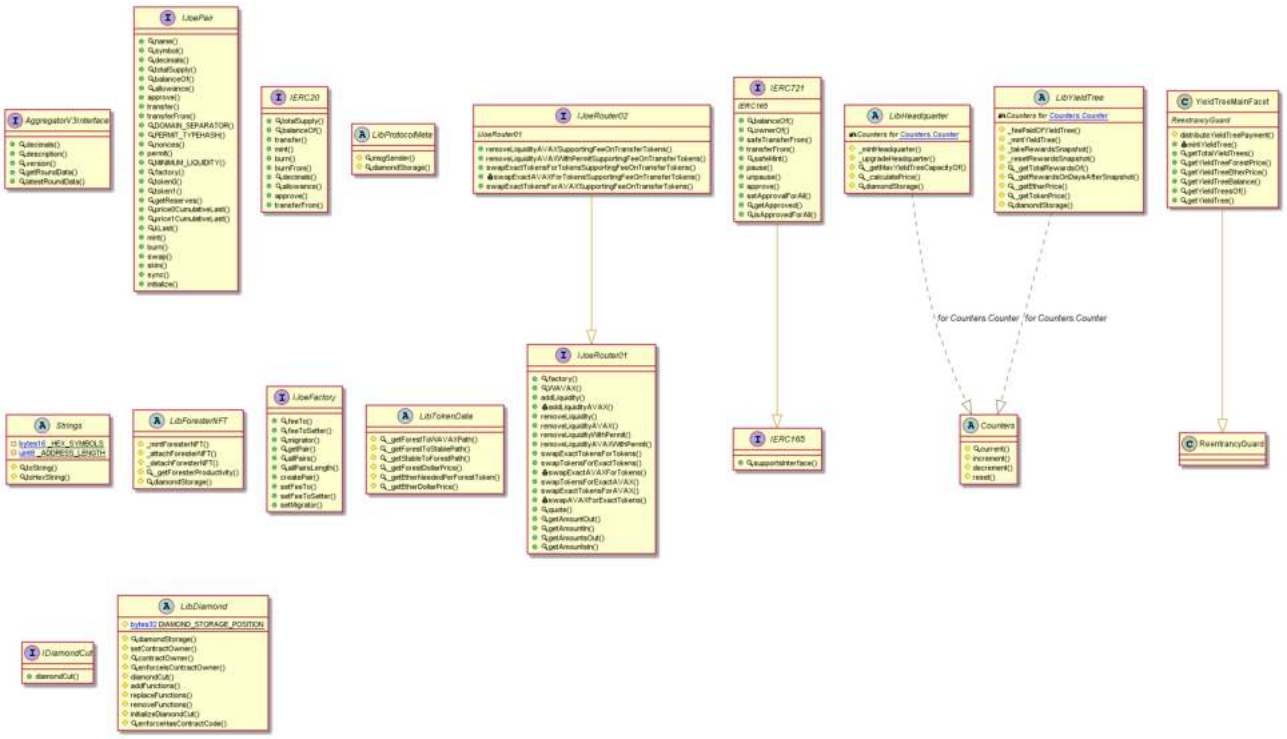
ProtocolDataManageFacet Diagram



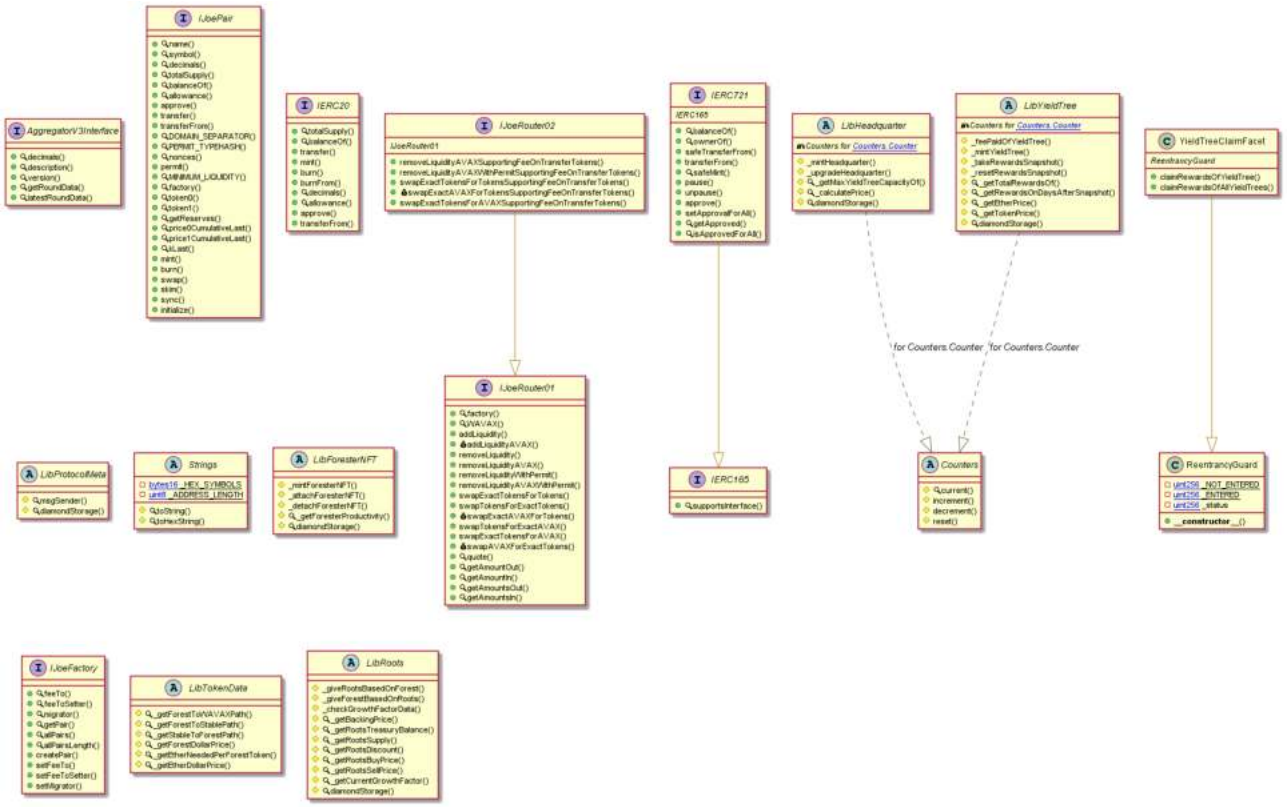
RootsMainFacet Diagram



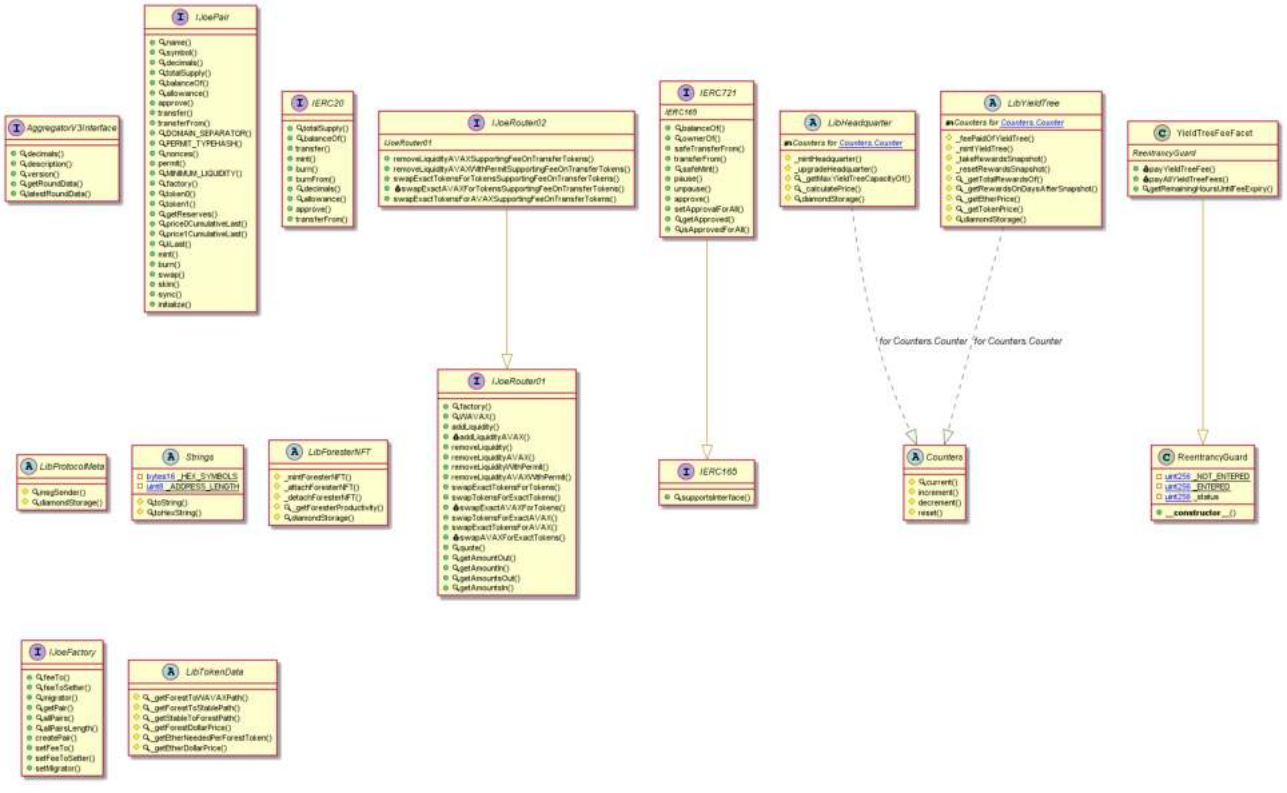
YieldTreeMainFacet Diagram



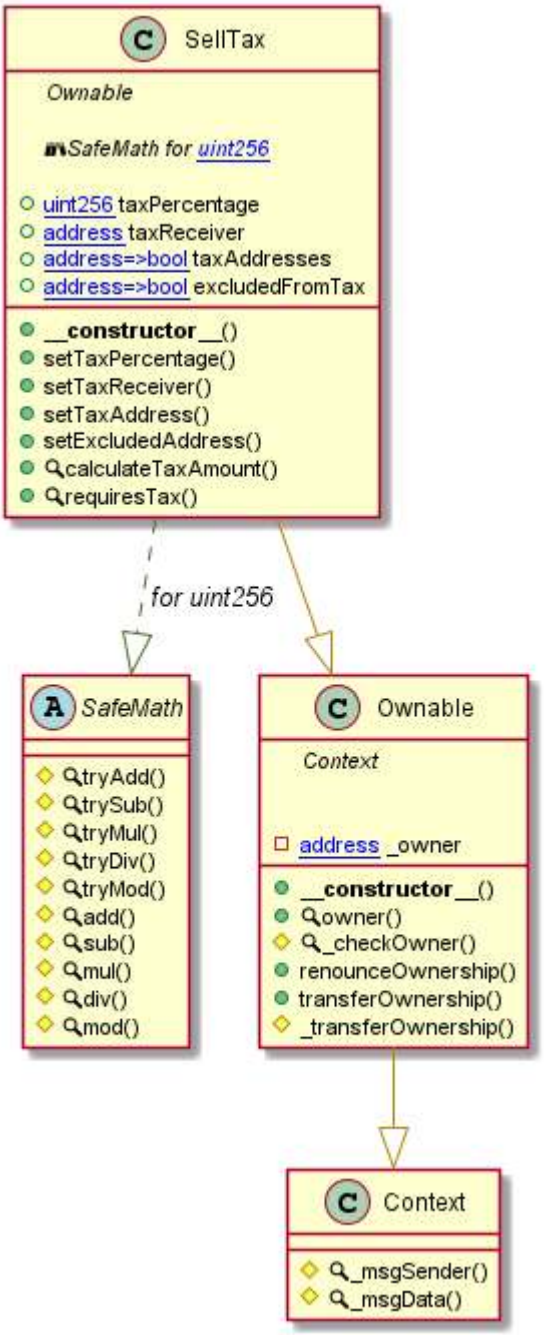
YieldTreeClaimFacet Diagram



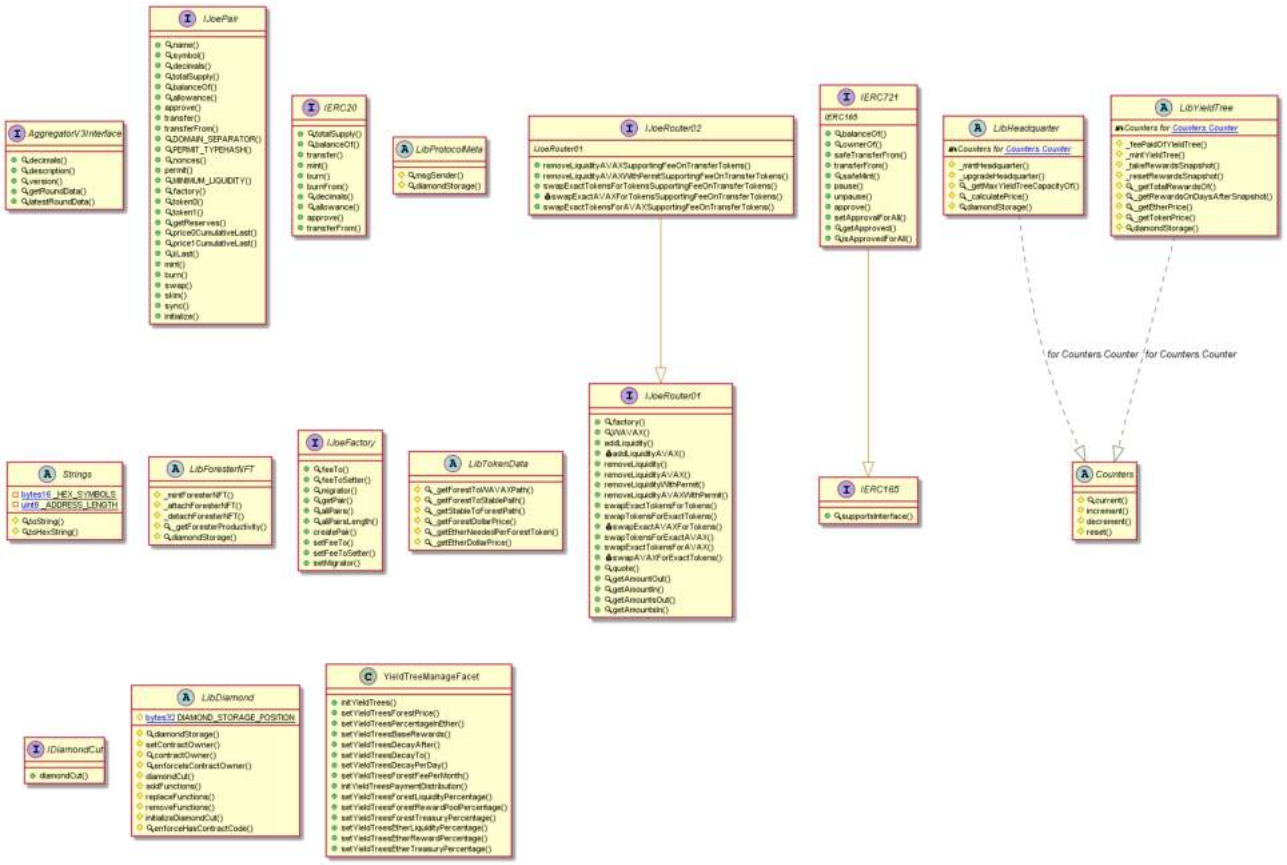
YieldTreeFeeFacet Diagram



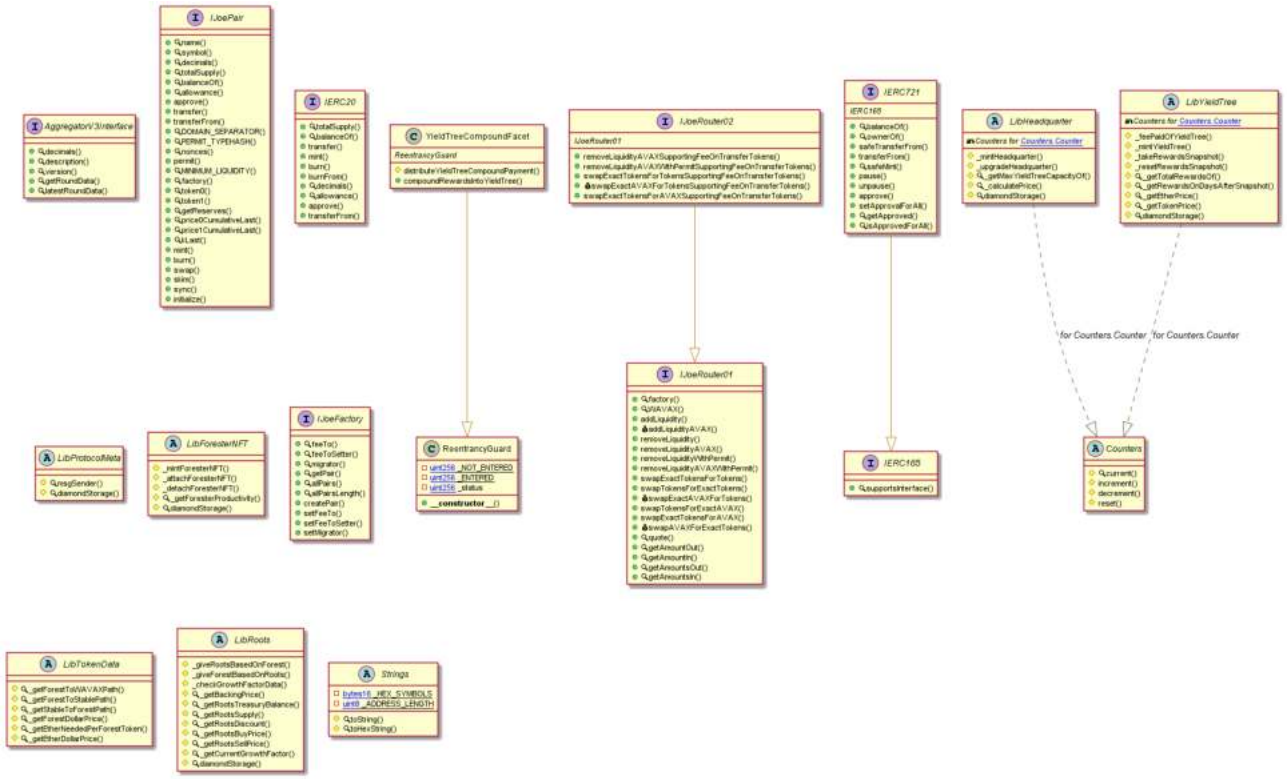
SellTax Diagram



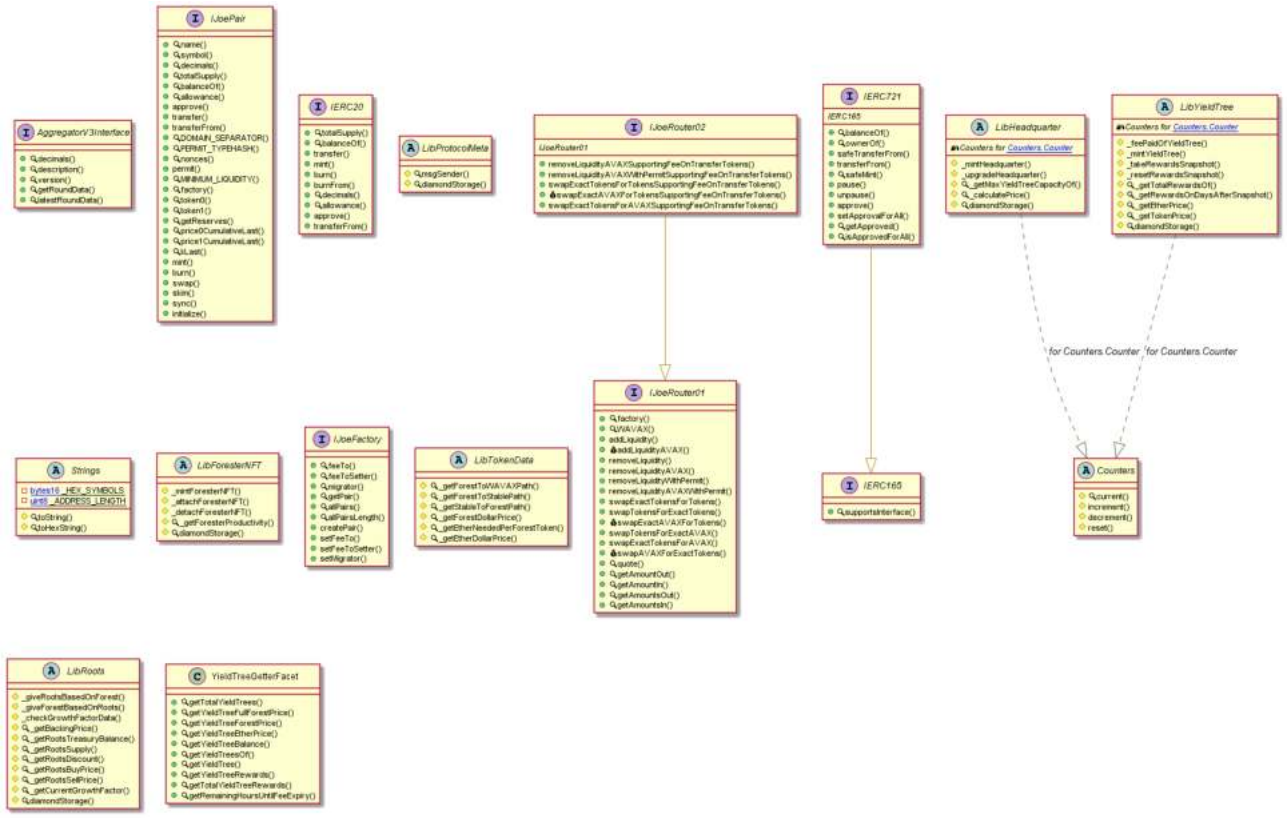
YieldTreeManageFacet Diagram



YieldTreeCompoundFacet Diagram



YieldTreeGetterFacet Diagram



Email: audit@EtherAuthority.io