

Mar 13, 2024



Security Assessment

hebao_v3

Professional Service

Table of Contents

1. Overview

- 1.1. [Executive Summary](#)
- 1.2. [Project Summary](#)
- 1.3. [Assessment Summary](#)
- 1.4. [Assessment Scope](#)

2. Checklist

3. Findings

- [M-01: Wallet Can Be Locked Permanently](#)
- [L-02: Using Trusted Price Oracle](#)
- [L-03: Use Safer Functions](#)
- [L-04: Clarify Return Value](#)
- [L-05: Initialization Can Be Front-run](#)
- [I-06: Unnecessary Approval](#)
- [I-07: Laggard Validity Check](#)
- [I-08: Unnecessary Experimental Pragma](#)
- [I-09: Use CustomError Instead of String](#)
- [I-10: No Check of Address Params with Zero Address](#)
- [I-11: Unused Imported Contract](#)
- [I-12: Unused Internal Function](#)
- [I-13: Parameters Should Be Declared as Calldata](#)
- [I-14: Variables Can Be Declared as Immutable](#)
- [I-15: No Need To Use SafeMath in Solidity Contract of Version 0.8.0 and Above](#)
- [I-16: Unused Events](#)
- [I-17: Set the Constant to Private](#)
- [I-18: Interface Defined Not Inherited](#)

4. Disclaimer

5. Appendix

1. Overview

1.1. Executive Summary

Hebao Smart Wallet Contracts is the smart contracts underpinning *Loopring's* advanced smart-wallet deployment.

The security assessment encompassed the `base` directory for project core logic, the `connectors` directory for various defi protocol integration and the `price` directory for onchain price oracle from other DEX protocols, alongside any contract dependencies external to officially accredited libraries. The remaining parts of the project fall outside the scope of current audit, as they are directly derived from other projects which have undergone audits by third-party entities.

Our comprehensive analysis employed a triad of methodologies: *Static Analysis*, *Formal Verification*, and *Manual Review*. Upon scrutinizing the contract, we identified **1 medium**, **4 low** and **13 informational** issues.

In commit 49167c6, **the project team has resolved the code security and business security issues stated in M-01, L-03, and L-04. Regarding the issues mentioned in L-02 and L-05, the project team has confirmed that there are no security risks, and therefore, no changes have been made.** In addition, the project team has partially applied the optimization suggestions.

1.2. Project Summary

| | |
|---------------------|---|
| Project Name | hebao_v3 |
| Platform | Ethereum |
| Language | Solidity |
| Codebase | Audit 1: <ul style="list-style-type: none">https://github.com/Loopring/protocols/tree/1a6952c522c95789fc276df29e83a9d21c83974a Final Audit: <ul style="list-style-type: none">https://github.com/Loopring/protocols/tree/49167c6b6748ac43f473dc321d8ebf62475a38c3 |

1.3. Assessment Summary

| | |
|--------------------------|---|
| Delivery Date | Mar 13, 2024 |
| Audit Methodology | Static Analysis, Formal Verification, Manual Review |

1.4. Assessment Scope

| ID | File | File Hash |
|-----------|---|----------------------------------|
| 1 | /hebao_v3/contracts/base/ConnectorRegistry.sol | f07731f732f513144c169ced4d984760 |
| 2 | /hebao_v3/contracts/base/DelayedImplementationManager.sol | 4a95aef3ed628a4eb26ea3bbb5254c0d |
| 3 | /hebao_v3/contracts/base/ForwardProxy.sol | 4b2cc3cb1f6d5395ec64c9cfdb155490 |
| 4 | /hebao_v3/contracts/base/libwallet/ApprovalLib.sol | dee6323747576b7a9f1b514a6f4f6cdc |
| 5 | /hebao_v3/contracts/base/libwallet/AutomationLib.sol | dc89ce1ca14cff9de82bddc79d3c9e85 |
| 6 | /hebao_v3/contracts/base/libwallet/ERC1271Lib.sol | bcb16dc0e1e3f2c3a461df8f58668535 |
| 7 | /hebao_v3/contracts/base/libwallet/ERC20Lib.sol | 5173ac2029eb9f75dc0dfe71c57ed981 |
| 8 | /hebao_v3/contracts/base/libwallet/GuardianLib.sol | f5adbd0dd18c2b73234eb1e20510c2fb |
| 9 | /hebao_v3/contracts/base/libwallet/InheritanceLib.sol | 373a00b97ce6b59e652257860faf5676 |
| 10 | /hebao_v3/contracts/base/libwallet/LockLib.sol | c33e5cf4f97af39bc808b3e65ab1eab |
| 11 | /hebao_v3/contracts/base/libwallet/QuotaLib.sol | f84074af6b6e23d6a1d0f90729d84aa9 |
| 12 | /hebao_v3/contracts/base/libwallet/RecoverLib.sol | e0a882b3b2daab51cb3b266d912495dc |
| 13 | /hebao_v3/contracts/base/libwallet/UpgradeLib.sol | fb318137b74ac0af5d48534f03a21f93 |
| 14 | /hebao_v3/contracts/base/libwallet/Utils.sol | aca248edfc447ecc9f319941f3b2ff3e |
| 15 | /hebao_v3/contracts/base/libwallet/WalletData.sol | dfaf71a9a4353d26b015cc4e02f72e2c |
| 16 | /hebao_v3/contracts/base/libwallet/WhitelistLib.sol | 9dd16fbba45b0f21dbb1a6999a447033 |

| ID | File | File Hash |
|-----------|--|----------------------------------|
| 17 | /hebao_v3/contracts/base/LoopringCreate2Deployer.sol | ee3f87e30f2654e03841dfa2dc0994a9 |
| 18 | /hebao_v3/contracts/base/LoopringPaymaster.sol | bd811ba963b69dad37e4cd54218a0ac2 |
| 19 | /hebao_v3/contracts/base/OfficialGuardian.sol | b1600a7cd89d9bd4c1884100fa17359f |
| 20 | /hebao_v3/contracts/base/SmartWallet.sol | 778d81c7d617184d39b6a22715723be4 |
| 21 | /hebao_v3/contracts/base/SmartWalletV3.sol | 7bfe48032453f8da810b8343e0cb2d35 |
| 22 | /hebao_v3/contracts/base/WalletDeploymentLib.sol | 2b79f54caa2b09d5e004281868281f53 |
| 23 | /hebao_v3/contracts/base/WalletFactory.sol | 4458717bce7288b638e21ce96df24130 |
| 24 | /hebao_v3/contracts/connectors/aavev3.sol | 961d304ab3ccfdd7d54356f174ae63f5 |
| 25 | /hebao_v3/contracts/connectors/base_connector.sol | c724b28ddf716a1ca604718dadfdd744 |
| 26 | /hebao_v3/contracts/connectors/compound.sol | be02f14ecb2e33856da696b90a271257 |
| 27 | /hebao_v3/contracts/connectors/flashloan/BalancerFlashloan.sol | ae3de20a4b89d9bf52990618e797ba34 |
| 28 | /hebao_v3/contracts/connectors/flashloan/flashloan_connector.sol | ce837e2e4bb0723323a2305bcec8cd00 |
| 29 | /hebao_v3/contracts/connectors/flashloan/IFlashLoanRecipient.sol | d4001230eff57d5834438904bae4352d |
| 30 | /hebao_v3/contracts/connectors/flashloan/IVault.sol | fdf8948b29cd4ac4e5fcdd6efcb9a3c8 |
| 31 | /hebao_v3/contracts/connectors/lido.sol | c9d82f5ebaf4174276c6cd6e2bc0abd5 |
| 32 | /hebao_v3/contracts/connectors/memory.sol | 329ded52e90711f77608d85b9a59b160 |
| 33 | /hebao_v3/contracts/connectors/uniswap.sol | e89ced41c1341f2f1e89943ccb52099e |

| ID | File | File Hash |
|----|--|----------------------------------|
| 34 | /hebao_v3/contracts/connectors/weth.sol | 32d4da50b4b61e5318130f6a16face83 |
| 35 | /hebao_v3/contracts/price/AggregationalPriceOracle.sol | 1cd109767d30dc526351eacac6c18989 |
| 36 | /hebao_v3/contracts/price/CachedPriceOracle.sol | 852c9533bfc9e349490f54caa6a7dc78 |
| 37 | /hebao_v3/contracts/price/KyberNetworkPriceOracle.sol | b502f6a11a3b24fd032125804754b2a5 |
| 38 | /hebao_v3/contracts/price/UniswapV2PriceOracle.sol | d6982295d884b080cb1d64423256060e |

2. Checklist

2.1. Code Security

| | | |
|----------------------------------|----------------------------------|---------------------------|
| Reentrancy | DelegateCall | Integer Overflow |
| Input Validation | Unchecked this.call | Frozen Money |
| Arbitrary External Call | Unchecked Owner Transfer | Do-while Continue |
| Right-To-Left-Override Character | Unauthenticated Storage Access | Risk For Weak Randomness |
| TxOrigin | Missing Checks for Return Values | Diamond Inheritance |
| ThisBalance | VarType Deduction | Array Length Manipulation |
| Uninitialized Variable | Shadow Variable | Divide Before Multiply |
| Affected by Compiler Bug | | |

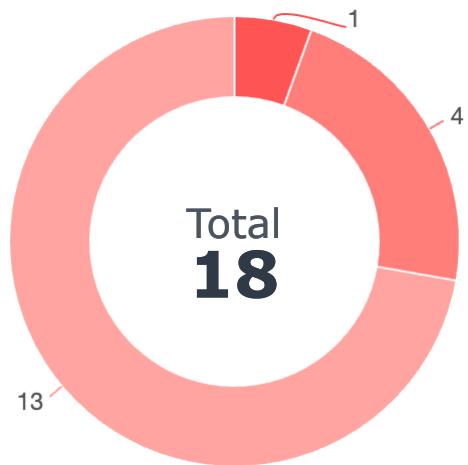
2.2. Optimization Suggestion

| | |
|--|--|
| Compiler Version | Improper State Variable Modification |
| Function Visibility | Deprecated Function |
| Externally Controlled Variables | Code Style |
| Constant Specific | Event Specific |
| Return Value Unspecified | Inexistent Error Message |
| State Variable Defined Without Storage Location | Import Issue |
| Compare With Timestamp/Block Number/Blockhash | Constructor in Base Contract Not Implemented |
| Delete Struct Containing the Mapping Type | Usage of '=+' |
| Paths in the Modifier Not End with "_" or Revert | Non-payable Public Functions Use msg.value |
| Lack of SafeMath | Compiler Error/Warning |
| Tautology Issue | Loop Depends on Array Length |
| Redundant/Duplicated/Dead Code | Code Complexity/Code Inefficiency |
| Undeclared Resource | Optimizable Return Statement |
| Unused Resource | |

2.3. Business Security

| |
|---|
| The Code Implementation is Consistent With Comments, Project White Papers and Other Materials |
| Permission Check |
| Address Check |

3. Findings



● Medium ● Low ● Informational

Code Security

Low: 2



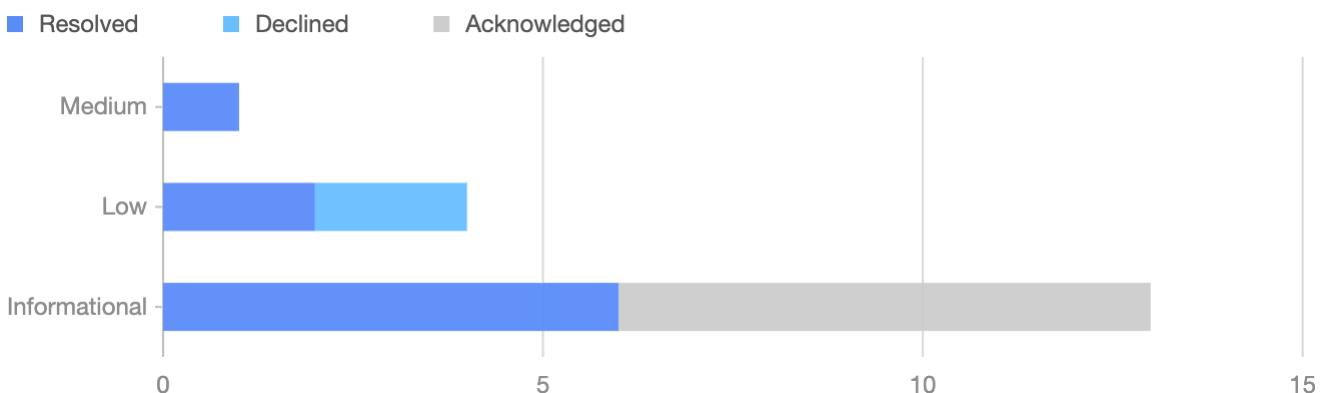
Optimization Suggestion

Low: 1, Informational: 13



Business Security

Medium: 1, Low: 1



| ID | Title | Category | Severity | Status |
|------|--|-------------------------|-----------------|--------------|
| M-01 | Wallet Can Be Locked Permanently | Business Security | ● Medium | Resolved |
| L-02 | Using Trusted Price Oracle | Business Security | ● Low | Declined |
| L-03 | Use Safer Functions | Code Security | ● Low | Resolved |
| L-04 | Clarify Return Value | Optimization Suggestion | ● Low | Resolved |
| L-05 | Initialization Can Be Front-run | Code Security | ● Low | Declined |
| I-06 | Unnecessary Approval | Optimization Suggestion | ● Informational | Resolved |
| I-07 | Laggard Validity Check | Optimization Suggestion | ● Informational | Resolved |
| I-08 | Unnecessary Experimental Pragma | Optimization Suggestion | ● Informational | Resolved |
| I-09 | Use CustomError Instead of String | Optimization Suggestion | ● Informational | Acknowledged |
| I-10 | No Check of Address Params with Zero Address | Optimization Suggestion | ● Informational | Acknowledged |

| ID | Title | Category | Severity | Status |
|-----------|---|-------------------------|-----------------|---------------|
| I-11 | Unused Imported Contract | Optimization Suggestion | ● Informational | Acknowledged |
| I-12 | Unused Internal Function | Optimization Suggestion | ● Informational | Acknowledged |
| I-13 | Parameters Should Be Declared as Calldata | Optimization Suggestion | ● Informational | Acknowledged |
| I-14 | Variables Can Be Declared as Immutable | Optimization Suggestion | ● Informational | Acknowledged |
| I-15 | No Need To Use SafeMath in Solidity Contract of Version 0.8.0 and Above | Optimization Suggestion | ● Informational | Resolved |
| I-16 | Unused Events | Optimization Suggestion | ● Informational | Resolved |
| I-17 | Set the Constant to Private | Optimization Suggestion | ● Informational | Acknowledged |
| I-18 | Interface Defined Not Inherited | Optimization Suggestion | ● Informational | Resolved |

M-01: Wallet Can Be Locked Permanently



Medium: Business Security

File Location: /hebao_v3/contracts/base/libwallet/GuardianLib.sol:119

Description

A wallet may become permanently locked due to the incorrect removal of the wallet's guardians.

Impact

The wallet can be locked permanently.

Proof of Concept

Here is a brief Proof of Concept (PoC):

1. The wallet's owner calls the `resetGuardians` function to clear the wallet's guardians.
2. The wallet's owner calls the `lock` function to lock the wallet.

At this point, unlocking the wallet must be done through the EntryPoint. However, the signature verification prior to the call will check if the invocation includes authorization from more than half of the guardians. Since the guardians have been cleared by the user at this stage, the call to the `requireMajority` function will fail. Therefore, the wallet will be permanently locked in this scenario. The only possibility for unlocking is if the user had previously set an inheritor and, after the waiting period ends, uses the inheritor to reset the owner and guardians.

/hebao_v3/contracts/base/libwallet/GuardianLib.sol

```
107      function requireMajority(
108          Wallet storage wallet,
109          address[] memory signers,
110          SigRequirement requirement
111      ) internal view returns (bool) {
112          // We always need at least one signer
113          if (signers.length == 0) {
114              return false;
115          }
116
117          // Calculate total group sizes
118          Guardian[] memory allGuardians = guardians(wallet, false);
119          _require(allGuardians.length > 0, Errors.NO_GUARDIANS);
```

Recommendation

When involving external calls to remove or modify Guardians, ensure that the wallet has at least one Guardian remaining after any such calls are completed.

Alleviation

Resolved in commit 49167c6. The project team addressed the issue by ensuring that the number of guardians is greater than zero when locking the wallet.

L-02: Using Trusted Price Oracle



Low: Business Security

File Location: /hebao_v3/contracts/price/UniswapV2PriceOracle.sol:24

Description

Liquidity pools based on a constant product formula, such as *Uniswap V2*, are susceptible to price manipulation through flash loans, leading to discrepancies in oracle outputs from actual market prices. This discrepancy might compromise the quota management features of wallets.

/hebao_v3/contracts/price/UniswapV2PriceOracle.sol

```
24     function tokenValue(
25         address token,
26         uint amount
27     ) public view override returns (uint) {
28         if (amount == 0) return 0;
29         if (token == address(0) || token == wethAddress) return amount;
30
31         address pair = factory.getPair(token, wethAddress);
32         if (pair == address(0)) {
33             return 0;
34         }
35
36         (uint112 reserve0, uint112 reserve1, ) = IUniswapV2Pair(pair)
37             .getReserves();
38
39         if (reserve0 == 0 || reserve1 == 0) {
40             return 0;
41         }
42
43         if (token < wethAddress) {
44             return amount.mul(reserve1) / reserve0;
45         } else {
46             return amount.mul(reserve0) / reserve1;
47         }
48     }
49 }
```

Recommendation

1. Avoid relying on easily manipulated data sources such as *Uniswap V2* pools directly as price oracles.
2. Calculate the *TWAP* (*Time-Weighted Average Price*) or use price oracles based on *TWAP*, e.g. *Uniswap V3*.

Alleviation

The project team confirmed that there is no security risk in the business logic, as the use of the quota is subject to strict permission controls. Therefore, no changes will be made to this issue.

L-03: Use Safer Functions

Low: Code Security



File Location:

/hebao_v3/contracts/connectors/aavev3.sol:173,277
/hebao_v3/contracts/connectors/base_connector.sol:124
/hebao_v3/contracts/connectors/compound.sol:105,224,266,378
/hebao_v3/contracts/connectors/uniswap.sol:186,247,299,300,337

Description

When calling the `transfer`, `transferFrom`, and `approve` functions in the ERC20 contract, there are some contracts that are not fully implemented in accordance with the ERC20 standard. In order to more comprehensively judge whether the call result meets expectations or to be compatible with different ERC20 contracts, it is recommended to use the `safeTransfer`, `safeTransferFrom`, `safeApprove` function.

/hebao_v3/contracts/connectors/aavev3.sol

```
171          }
172
173      tokenContract.approve(address(aave), _amt);
174
175      aave.supply(_token, _amt, address(this), REFERRAL_CODE);
```

/hebao_v3/contracts/connectors/aavev3.sol

```
275      if (isEth) convertEthToWeth(isEth, tokenContract, _amt);
276
277      tokenContract.approve(address(aave), _amt);
278
279      aave.repay(_token, _amt, rateMode, address(this));
```

/hebao_v3/contracts/connectors/base_connector.sol

```
122      ) internal {
123          if (isEth) {
124              token.approve(address(token), amount);
125              token.withdraw(amount);
126          }
```

/hebao_v3/contracts/connectors/compound.sol

```
103          ? tokenContract.balanceOf(address(this))
104          : _amt;
105      tokenContract.approve(cToken, _amt);
106      require(CTokenInterface(cToken).mint(_amt) == 0,
107              "deposit-failed");
108  }
```

/hebao_v3/contracts/connectors/compound.sol

```
222             "not-enough-token"
223         );
224     tokenContract.approve(cToken, _amt);
225     require(cTokenContract.repayBorrow(_amt) == 0, "repay-failed.
226         ");
227 }
```

/hebao_v3/contracts/connectors/compound.sol

```
264             ? tokenContract.balanceOf(address(this))
265             : _amt;
266     tokenContract.approve(cToken, _amt);
267     require(ctokenContract.mint(_amt) == 0,
268         "deposit-ctoken-failed.");
269 }
```

/hebao_v3/contracts/connectors/compound.sol

```
376             "not-enough-token"
377         );
378     tokenContract.approve(cTokenPay, _amt);
379     require(
380         cTokenContract.liquidateBorrow(borrower, _amt,
381         cTokenColl) == 0,
```

/hebao_v3/contracts/connectors/uniswap.sol

```
184     bool isEth = address(_sellAddr) == WETH_ADDR;
185     convertEthToWeth(isEth, _sellAddr, _expectedAmt);
186     _sellAddr.approve(address(ROUTER), _expectedAmt);
187
188     uint _sellAmt = ROUTER.swapTokensForExactTokens(
```

/hebao_v3/contracts/connectors/uniswap.sol

```
245     bool isEth = address(_sellAddr) == WETH_ADDR;
246     convertEthToWeth(isEth, _sellAddr, _sellAmt);
247     _sellAddr.approve(address(ROUTER), _sellAmt);
248
249     uint _buyAmt = ROUTER.swapExactTokensForTokens(
```

/hebao_v3/contracts/connectors/uniswap.sol

```
297     convertEthToWeth(isEth, _tokenB, _amtB);
298
299     _tokenA.approve(address(ROUTER), _amtA);
300     _tokenA.approve(address(ROUTER), _amtB);
301 }
```

/hebao_v3/contracts/connectors/uniswap.sol

```
298         _tokenA.approve(address(ROUTER), _amtA);
299         _tokenA.approve(address(ROUTER), _amtB);
300
301     uint minAmtA = getMinAmount(_tokenA, _amtA, slippage);
```

/hebao_v3/contracts/connectors/uniswap.sol

```
335             ? uniToken.balanceOf(address(this))
336             : _amt;
337         uniToken.approve(address(ROUTER), _uniAmt);
338
339     {
```

Recommendation

It is recommended to use the `safeTransfer`, `safeTransferFrom`, and `safeApprove` functions in SafeERC20 to call the `transfer`, `transferFrom`, and `approve` functions in the ERC20 contract.

Alleviation

Resolved in commit 49167c6.

L-04: Clarify Return Value

Low: Optimization Suggestion



File Location:

/hebao_v3/contracts/base/LoopringPaymaster.sol:164,176
/hebao_v3/contracts/base/SmartWallet.sol:415

Description

In function `_validatePaymasterUserOp` of `LoopringPaymaster` contract and function `callContractWA` of `SmartWallet` contract, the returned variable is specified in the function signature, but it still calls the return statement to return a local variable defined in the function body or state variable. It is necessary to clarify whether the returned value meets expectations.

/hebao_v3/contracts/base/LoopringPaymaster.sol

```
162      //don't revert on signature failure: return SIG_VALIDATION_FAILED
163      if (!hasRole(SIGNER, hash.recover(signature))) {
164          return (
165              "",
166              _packValidationData(
```

/hebao_v3/contracts/base/LoopringPaymaster.sol

```
174      //no need for other on-chain validation: entire UserOp should
175      // have been checked
176      // by the external service prior to signing it.
177      return (
178          abi.encode(
179              userOpHash,
```

/hebao_v3/contracts/base/SmartWallet.sol

```
413      bytes calldata data
414      ) external onlyFromEntryPoint returns (bytes memory returnData) {
415          return ERC20Lib.callContractWA(to, value, data);
416      }
417
```

Recommendation

It is recommended to be clear whether the returned value is as expected, and only use one way to return the value.

Alleviation

Resolved in commit 49167c6.

L-05: Initialization Can Be Front-run



Low: Code Security

File Location: /hebao_v3/contracts/base/OfficialGuardian.sol:16

Description

The function `initOwner` of `OfficialGuardian` contract has no access control. A malicious attacker could front-run the `initOwner` transaction to gain ownership of the contract. This can be repeated as a Denial Of Service (DOS) type of attack, effectively preventing contract deployment, leading to unrecoverable gas expenses.

/hebao_v3/contracts/base/OfficialGuardian.sol

```
15      /// @dev init owner for proxy contract:  
16      function initOwner(address _owner) external {  
17          require(owner == address(0), "INITIALIZED_ALREADY");  
18          owner = _owner;  
19      }
```

Recommendation

Ensure the deployment and initialization of contracts are in the same transaction or add `onlyOwner` or other authentication checking in initialize function.

Alleviation

The project team confirmed that the permission check for the `initOwner` function is implemented in the proxy contract. Therefore, no changes will be made to this issue.

I-06: Unnecessary Approval



Informational: Optimization Suggestion

File Location: /hebao_v3/contracts/connectors/weth.sol:26

Description

Obtaining approval prior to extracting *ETH* from *WETH* is unnecessary.

/hebao_v3/contracts/connectors/weth.sol

```
20      function withdraw(uint amt, uint getId, uint setId) public payable {
21          uint _amt = getUint(getId, amt);
22
23          _amt = _amt == type(uint256).max
24              ? WETH_CONTRACT.balanceOf(address(this))
25              : _amt;
26          WETH_CONTRACT.approve(WETH_ADDR, _amt);
27          WETH_CONTRACT.withdraw(_amt);
28          setUint(setId, _amt);
29      }
```

Recommendation

Eliminate the redundant invocation of the function `approve`.

Alleviation

Resolved in commit 49167c6. The unnecessary approval is removed.

I-07: Laggard Validity Check



Informational: Optimization Suggestion

File Location: /hebao_v3/contracts/base/LoopringPaymaster.sol:297

Description

Positioning validity checks at the commencement of functions can save gas in the event of failure.

/hebao_v3/contracts/base/LoopringPaymaster.sol

```
290     function addDepositFor(
291         address token,
292         address account,
293         uint256 amount
294     ) external {
295         // (sender must have approval for the paymaster)
296         IERC20(token).safeTransferFrom(msg.sender, address(this),
297             amount);
297         require(registeredToken[token], "LoopringPaymaster: unsupported
298             token");
298         balances[token][account] += amount;
299         if (msg.sender == account) {
300             lockTokenDeposit();
301         }
302     }
```

Recommendation

Relocate all validity checks to the foremost positions within functions wherever feasible.

Alleviation

Resolved in commit 49167c6. The project team relocates the validity check to the beginning of function `addDepositFor`.

I-08: Unnecessary Experimental Pragma



Informational: Optimization Suggestion

File Location:

Description

For contracts utilizing *Solidity version 0.7.4* and subsequent versions, the inclusion of the experimental pragma `pragma experimental ABIEncoderV2;` to activate the `ABI coder v2` is rendered unnecessary.

Recommendation

Eliminate all instances of the experimental pragma `pragma experimental ABIEncoderV2;` within the contracts.

Alleviation

Resolved in commit 49167c6. The unnecessary experimental pragma is removed.

I-09: Use CustomError Instead of String



Informational: Optimization Suggestion

File Location:

Description

When using `require` or `revert`, `CustomError` is more gas efficient than string description, as the error message described using `CustomError` is only compiled into four bytes.

Especially when string exceeds 32 bytes, more gas will be consumed. Generally, around 250-270 gas can be saved for one `CustomError` replacement when compiler optimization is turned off, 60-80 gas can be saved even if compiler optimization is turned on.

This issue is prevalent in almost all contracts that use `require` and `revert` statements.

Recommendation

Use `CustomError` instead of string for `require` or `revert` description.

Alleviation

Acknowledged and partially applied in commit 49167c6. To ensure that external `EntryPoints` can correctly parse error messages returned by `LoopringPaymaster`, some checks still use revert strings instead of custom errors.

I-10: No Check of Address Params with Zero Address

Informational: Optimization Suggestion

File Location:

/hebao_v3/contracts/base/LoopringPaymaster.sol:49,50,270,275
/hebao_v3/contracts/base/OfficialGuardian.sol:16,31
/hebao_v3/contracts/base/SmartWallet.sol:117,118,119,120,341,347,352,491,498
/hebao_v3/contracts/base/SmartWalletV3.sol:69
/hebao_v3/contracts/base/WalletDeploymentLib.sol:17
/hebao_v3/contracts/base/WalletFactory.sol:57,63
/hebao_v3/contracts/base/libwallet/WhitelistLib.sol:27,34,38
/hebao_v3/contracts/connectors/base_connector.sol:83
/hebao_v3/contracts/connectors/flashloan/BalancerFlashloan.sol:33
/hebao_v3/contracts/connectors/flashloan/flashloan_connector.sol:25,26
/hebao_v3/contracts/connectors/memory.sol:62,102



Description

The input parameter of the `address` type in the function does not use the zero address for verification. This might lead to loss of fund, particularly in scenarios where low-level calls to the zero address will not revert.

/hebao_v3/contracts/base/LoopringPaymaster.sol

```
48     constructor(  
49         IEntryPoint _entryPoint,  
50         address paymasterOwner
```

/hebao_v3/contracts/base/LoopringPaymaster.sol

```
270     function addToken(address token) external onlyOwner {  
271         require(!registeredToken[token], "registered already");  
272         registeredToken[token] = true;  
273     }  
274     function removeToken(address token) external onlyOwner {  
275         require(registeredToken[token], "unregistered already");  
276         registeredToken[token] = false;  
277     }
```

/hebao_v3/contracts/base/OfficialGuardian.sol

```
16     function initOwner(address _owner) external {  
17         require(owner == address(0), "INITIALIZED_ALREADY");  
18         owner = _owner;
```

/hebao_v3/contracts/base/OfficialGuardian.sol

```
31     function transact(
32         address target,
33         uint value,
```

/hebao_v3/contracts/base/SmartWallet.sol

```
116     constructor(
117         PriceOracle _priceOracle,
118         address _blankOwner,
119         IEntryPoint entryPointInput,
120         address _connectorRegistry
121     ) {
```

/hebao_v3/contracts/base/SmartWallet.sol

```
341     function addToWhitelist(
342         address addr
343     ) external onlyFromEntryPointOrWalletOrOwnerWhenUnlocked {
344         wallet.addToWhitelist(addr);
345     }
346
347     function addToWhitelistWA(address addr) external onlyFromEntryPoint {
348         wallet.addToWhitelistWA(addr);
349     }
350
351     function removeFromWhitelist(
352         address addr
353     ) external onlyFromEntryPointOrWalletOrOwnerWhenUnlocked {
354         wallet.removeFromWhitelist(addr);
355     }
```

/hebao_v3/contracts/base/SmartWallet.sol

```
490     function approveExecutor(
491         address executor,
492         uint256 validUntil
493     ) external onlyFromEntryPointOrWalletOrOwnerWhenUnlocked {
494         return AutomationLib.approveExecutor(wallet, executor,
495             validUntil);
496     }
497
498     function unApproveExecutor(
499         address executor
500     ) external onlyFromEntryPointOrWalletOrOwnerWhenUnlocked {
501         return AutomationLib.unApproveExecutor(wallet, executor);
```

/hebao_v3/contracts/base/SmartWalletV3.sol

```
68     function withdrawDepositTo(
69         address payable withdrawAddress,
70         uint256 amount
```

/hebao_v3/contracts/base/WalletDeploymentLib.sol

```
17     constructor(address _walletImplementation) {
18         walletImplementation = _walletImplementation;
19     }
```

/hebao_v3/contracts/base/WalletFactory.sol

```
55     /// @dev Adds a new operator.
56     /// @param operator The new address to add.
57     function addOperator(address operator) public onlyOwner {
58         addOperatorInternal(operator);
59     }
60
61     /// @dev Removes a operator.
62     /// @param operator The operator to remove.
63     function removeOperator(address operator) public onlyOwner {
64         removeAddressFromSet(OPERATOR, operator);
65         emit OperatorRemoved(operator);
66     }
```

/hebao_v3/contracts/base/libwallet/WhitelistLib.sol

```
27     function addToWhitelist(Wallet storage wallet, address addr)
28         external {
29         wallet._addToWhitelist(
30             addr,
31             block.timestamp.add(WHITELIST_PENDING_PERIOD)
32         );
33
34     function addToWhitelistWA(Wallet storage wallet, address addr)
35         external {
36         wallet._addToWhitelist(addr, block.timestamp);
37
38     function removeFromWhitelist(Wallet storage wallet, address addr)
39         external {
40         wallet._removeFromWhitelist(addr);
```

```
/hebao_v3/contracts/connectors/base_connector.sol
```

```
83     constructor(address _instaMemory) {
84         instaMemory = MemoryInterface(_instaMemory);
85     }
```

```
/hebao_v3/contracts/connectors/flashloan/BalancerFlashloan.sol
```

```
33     constructor(address _vault) {
34         vault = _vault;
35     }
```

```
/hebao_v3/contracts/connectors/flashloan/flashloan_connector.sol
```

```
24     constructor(
25         address _instaMemory,
26         address _flashLoanPool
```

```
/hebao_v3/contracts/connectors/memory.sol
```

```
62     function setAddr(uint _id, address _addr) public {
63         maddr[msg.sender][_id] = _addr;
64     }
```

```
/hebao_v3/contracts/connectors/memory.sol
```

```
102    function setBroadcastAddr(uint _id, address _addr) public isMaster {
103        maddr[BROADCAST_ADDR][_id] = _addr;
104    }
```

Recommendation

It is recommended to perform zero address verification on the input parameters of the `address` type. Alternatively, verify the transaction calldata and result meticulously when dealing with privileged functions, such as the `initialize` function, `constructor`, and similar critical functions.

Alleviation

Acknowledged and partially applied in commit 49167c6. Functions without a zero-address check can only be called by privileged accounts or will not affect the contract's execution logic.

I-11: Unused Imported Contract

Informational: Optimization Suggestion



File Location:
/hebao_v3/contracts/base/SmartWallet.sol:12
/hebao_v3/contracts/base/libwallet/Utils.sol:6

Description

Unused imported contract will increase gas consumption.

/hebao_v3/contracts/base/SmartWallet.sol

```
10 import "../lib/EIP712.sol";
11 import "../lib/ERC1271.sol";
12 import "@openzeppelin/contracts/security/ReentrancyGuard.sol";
13 import "../thirdparty/erc165/IERC165.sol";
14 import "../thirdparty/erc1155/ERC1155Holder.sol";
```

/hebao_v3/contracts/base/libwallet/Utils.sol

```
4 pragma experimental ABIEncoderV2;
5
6 import "./WalletData.sol";
7 import "../../lib/AddressUtil.sol";
8
```

Recommendation

It is recommended to delete unused imported contracts.

Alleviation

Resolved in commit 49167c6.

I-12: Unused Internal Function



Informational: Optimization Suggestion

File Location:

Description

The following internal functions in `DSMath` contract is defined but not used, which will add gas consumption:

- `function add(uint x, uint y)`
- `function div(uint x, uint y)`
- `function wdiv(uint x, uint y)`
- `function rdiv(uint x, uint y)`
- `function rmul(uint x, uint y)`
- `function toInt(uint x)`
- `function toUInt(int256 x)`
- `function toRad(uint wad)`

Recommendation

It is recommended to delete functions that are not called.

Alleviation

Acknowledged and partially applied in commit 49167c6.

I-13: Parameters Should Be Declared as Calldata

Informational: Optimization Suggestion

File Location:

/hebao_v3/contracts/base/LoopringCreate2Deployer.sol:9
/hebao_v3/contracts/base/OfficialGuardian.sol:23
/hebao_v3/contracts/base/SmartWallet.sol:193
/hebao_v3/contracts/base/libwallet/ERC1271Lib.sol:22
/hebao_v3/contracts/base/libwallet/ERC20Lib.sol:237,261,286,319
/hebao_v3/contracts/base/libwallet/GuardianLib.sol:44,445,465,485
 /hebao_v3/contracts/base/libwallet/LockLib.sol:46
/hebao_v3/contracts/base/libwallet/RecoverLib.sol:67
/hebao_v3/contracts/base/libwallet/UpgradeLib.sol:32
/hebao_v3/contracts/base/libwallet/WhitelistLib.sol:74
/hebao_v3/contracts/connectors/flashloan/BalancerFlashloan.sol:17,38,39,40,41,70
/hebao_v3/contracts/connectors/flashloan/IFlashLoanRecipient.sol:32,33,34,35
/hebao_v3/contracts/connectors/flashloan/IVault.sol:10,11,12
/hebao_v3/contracts/connectors/flashloan_flashloan_connector.sol:40

Description

When the compiler parses the `external` or `public` function, it can directly read the function parameters from calldata. Setting it to other storage locations may waste gas. About 300-400 gas can be saved with optimization turned off while 120-150 gas can be saved vice versa.

/hebao_v3/contracts/base/LoopringCreate2Deployer.sol

```
7     event Deployed(address addr, uint256 salt);
8
9     function deploy(bytes memory code, uint256 salt) public {
10        address addr;
11        // solhint-disable-next-line no-inline-assembly
```

/hebao_v3/contracts/base/OfficialGuardian.sol

```
21     function isValidSignature(
22         bytes32 _signHash,
23         bytes memory _signature
24     ) public view override returns (bytes4) {
25         return
```

/hebao_v3/contracts/base/SmartWallet.sol

```
191    function isValidSignature(
192        bytes32 signHash,
193        bytes memory signature
194    ) public view override returns (bytes4 magicValue) {
195        return wallet.isValidSignature(ERC1271_MAGICVALUE, signHash,
196        signature);
```

/hebao_v3/contracts/base/libwallet/ERC1271Lib.sol

```
20         bytes4 erc1271MagicValue,
21         bytes32 signHash,
22         bytes memory signature
23     ) public view returns (bytes4 magicValue) {
24         if (wallet.locked) {
```

/hebao_v3/contracts/base/libwallet/ERC20Lib.sol

```
235
236     function encodeApprovalForTransferToken(
237         bytes memory data,
238         bytes32 domainSeparator,
239         uint256 validUntil
```

/hebao_v3/contracts/base/libwallet/ERC20Lib.sol

```
259
260     function encodeApprovalForApproveToken(
261         bytes memory data,
262         bytes32 domainSeparator,
263         uint256 validUntil
```

/hebao_v3/contracts/base/libwallet/ERC20Lib.sol

```
284
285     function encodeApprovalForCallContract(
286         bytes memory callData,
287         bytes32 domainSeparator,
288         uint256 validUntil
```

/hebao_v3/contracts/base/libwallet/ERC20Lib.sol

```
317
318     function encodeApprovalForApproveThenCallContract(
319         bytes memory callData,
320         bytes32 domainSeparator,
321         uint256 validUntil
```

/hebao_v3/contracts/base/libwallet/GuardianLib.sol

```
42     function addGuardiansImmediately(
43         Wallet storage wallet,
44         address[] memory _guardians
45     ) external {
46         address guardian = address(0);
```

/hebao_v3/contracts/base/libwallet/GuardianLib.sol

```
443
444     function encodeApprovalForAddGuardian(
445         bytes memory data,
446         bytes32 domainSeparator,
447         uint256 validUntil
```

/hebao_v3/contracts/base/libwallet/GuardianLib.sol

```
463
464     function encodeApprovalForRemoveGuardian(
465         bytes memory data,
466         bytes32 domainSeparator,
467         uint256 validUntil
```

/hebao_v3/contracts/base/libwallet/GuardianLib.sol

```
483
484     function encodeApprovalForResetGuardians(
485         bytes memory data,
486         bytes32 domainSeparator,
487         uint256 validUntil
```

/hebao_v3/contracts/base/libwallet/LockLib.sol

```
44
45     function encodeApprovalForUnlock(
46         bytes memory,
47         bytes32 domainSeparator,
48         uint256 validUntil
```

/hebao_v3/contracts/base/libwallet/RecoverLib.sol

```
65     function encodeApprovalForRecover(
66         address newOwner,
67         address[] memory newGuardians,
68         bytes32 domainSeparator,
69         uint256 validUntil
```

/hebao_v3/contracts/base/libwallet/UpgradeLib.sol

```
30
31     function encodeApprovalForChangeMasterCopy(
32         bytes memory data,
33         bytes32 domainSeparator,
34         uint256 validUntil
```

/hebao_v3/contracts/base/libwallet/WhitelistLib.sol

```
72
73     function encodeApprovalForAddToWhitelist(
74         bytes memory data,
75         bytes32 domainSeparator,
76         uint256 validUntil
```

/hebao_v3/contracts/connectors/flashloan/BalancerFlashloan.sol

```
15         address token,
16         uint256 amount,
17         bytes memory userData
18     ) external;
19 }
```

/hebao_v3/contracts/connectors/flashloan/BalancerFlashloan.sol

```
36
37     function receiveFlashLoan(
38         IERC20[] memory tokens,
39         uint256[] memory amounts,
40         uint256[] memory /*feeAmounts*/,
41         bytes memory userData
42     ) external override {
43         // check msg.sender
```

/hebao_v3/contracts/connectors/flashloan/BalancerFlashloan.sol

```
68         address token,
69         uint256 amount,
70         bytes memory data
71     ) external {
72         IERC20[] memory tokens = new IERC20[](1);
```

/hebao_v3/contracts/connectors/flashloan/IFlashLoanRecipient.sol

```
30     */
31     function receiveFlashLoan(
32         IERC20[] memory tokens,
33         uint256[] memory amounts,
34         uint256[] memory feeAmounts,
35         bytes memory userData
36     ) external;
37 }
```

/hebao_v3/contracts/connectors/flashloan/IVault.sol

```
8     function flashLoan(
9         IFlashLoanRecipient recipient,
10        IERC20[] memory tokens,
11        uint256[] memory amounts,
12        bytes memory userData
13    ) external;
14 }
```

/hebao_v3/contracts/connectors/flashloan/flashloan_connector.sol

```
38         address token,
39         uint amt,
40         bytes memory data
41     ) external payable {
42         (address[] memory targets, bytes[] memory callDatas) = abi.decode
43         (
```

Recommendation

In `external` or `public` functions, the storage location of function parameters should be set to `calldata` to save gas.

Alleviation

Acknowledged and partially applied in commit 49167c6.

I-14: Variables Can Be Declared as Immutable

Informational: Optimization Suggestion



File Location:
/hebao_v3/contracts/base/SmartWallet.sol:64
/hebao_v3/contracts/connectors/memory.sol:77
/hebao_v3/contracts/price/UniswapV2PriceOracle.sol:15

Description

The solidity compiler of version 0.6.5 introduces `immutable` to modify state variables that are only modified in the constructor. Using `immutable` can save gas.

/hebao_v3/contracts/base/SmartWallet.sol

```
62     address internal masterCopy;
63
64     bool internal isImplementationContract;
65
66     Wallet public wallet;
```

/hebao_v3/contracts/connectors/memory.sol

```
75
76     contract OwnedMemory is Memory {
77         address public master;
78         address public constant BROADCAST_ADDR = address(0);
79     }
```

/hebao_v3/contracts/price/UniswapV2PriceOracle.sol

```
13     using SafeMath for uint;
14
15     IUniswapV2Factory public factory;
16     address public immutable wethAddress;
17 
```

Recommendation

For contracts compiled with compiler of versions 0.6.5 and above, if the state variable is only modified in the constructor, it is recommended to modify the variable with `immutable` to save gas.

Alleviation

Acknowledged and partially applied in commit 49167c6.

I-15: No Need To Use SafeMath in Solidity Contract of Version 0.8.0 and Above

Informational: Optimization Suggestion



File Location:
/hebao_v3/contracts/base/libwallet/ERC20Lib.sol:19
/hebao_v3/contracts/base/libwallet/QuotaLib.sol:17
/hebao_v3/contracts/price/UniswapV2PriceOracle.sol:12

Description

In solidity 0.8.0 and above, the compiler has its own overflow checking function, so there is no need to use the SafeMath library to prevent overflow.

/hebao_v3/contracts/base/libwallet/ERC20Lib.sol

```
17 /// @author Brecht Devos - <brecht@loopring.org>
18 /// @author Daniel Wang - <daniel@loopring.org>
19 library ERC20Lib {
20     using AddressUtil for address;
21     using BytesUtil for bytes;
```

/hebao_v3/contracts/base/libwallet/QuotaLib.sol

```
15 /// @dev This store maintains daily spending quota for each wallet.
16 ///      A rolling daily limit is used.
17 library QuotaLib {
18     using SafeCast for uint;
19     using SafeMath for uint;
```

/hebao_v3/contracts/price/UniswapV2PriceOracle.sol

```
10 /// @title Uniswap2PriceOracle
11 /// @dev Returns the value in Ether for any given ERC20 token.
12 contract UniswapV2PriceOracle is PriceOracle {
13     using SafeMath for uint;
14
```

Recommendation

It is recommended to abandon the use of the SafeMath library to prevent overflow, which can save gas in versions 0.8.0 and above, unless you want to specify the error message when overflowing or you are using the openzeppelin library of version 0.4 and above.

Alleviation

Resolved in commit 49167c6.

I-16: Unused Events



Informational: Optimization Suggestion

File Location: /hebao_v3/contracts/base/libwallet/InheritanceLib.sol:24,35

Description

The `InheritorChanged` event is currently not being used. An event notification for `InheritorChanged` should be emitted when the `setInheritor` function is called, to allow off-chain tools to monitor changes to the inheritor.

/hebao_v3/contracts/base/libwallet/InheritanceLib.sol

```
22     event Inherited(address inheritor, address newOwner);
23
24     event InheritorChanged(address inheritor, uint32 waitingPeriod);
25
26     function touchLastActiveWhenRequired(Wallet storage wallet) internal
27     {
```

/hebao_v3/contracts/base/libwallet/InheritanceLib.sol

```
35     function setInheritor(
36         Wallet storage wallet,
37         address inheritor,
38         uint32 waitingPeriod
39     ) internal {
40         if (inheritor == address(0)) {
41             require(waitingPeriod == 0, "INVALID_WAITING_PERIOD");
42         } else {
43             require(
44                 waitingPeriod >= TOUCH_GRACE_PERIOD,
45                 "INVALID_WAITING_PERIOD"
46             );
47         }
48
49         require(inheritor != address(this), "INVALID_ARGS");
50         wallet.inheritor = inheritor;
51         wallet.inheritWaitingPeriod = waitingPeriod;
52         wallet.lastActive = uint64(block.timestamp);
53     }
```

Recommendation

It is recommended to emit the `InheritorChanged` event when the `setInheritor` function is called.

Alleviation

Resolved in commit 49167c6.

I-17: Set the Constant to Private

Informational: Optimization Suggestion

File Location:

/hebao_v3/contracts/base/ConnectorRegistry.sol:10
/hebao_v3/contracts/base/LoopringPaymaster.sol:34
/hebao_v3/contracts/base/WalletDeploymentLib.sol:15
/hebao_v3/contracts/base/WalletFactory.sol:28
/hebao_v3/contracts/base/libwallet/ERC20Lib.sol:34,38,42,46
/hebao_v3/contracts/base/libwallet/GuardianLib.sol:21,22,26,30,34
/hebao_v3/contracts/base/libwallet/InheritanceLib.sol:20
/hebao_v3/contracts/base/libwallet/LockLib.sol:22
/hebao_v3/contracts/base/libwallet/QuotaLib.sol:21,22,26
/hebao_v3/contracts/base/libwallet/RecoverLib.sol:24
/hebao_v3/contracts/base/libwallet/UpgradeLib.sol:20
/hebao_v3/contracts/base/libwallet/WhitelistLib.sol:16,20
/hebao_v3/contracts/connectors/base_connector.sol:24,25
/hebao_v3/contracts/connectors/memory.sol:78
/hebao_v3/contracts/price/CachedPriceOracle.sol:15
/hebao_v3/contracts/price/KyberNetworkPriceOracle.sol:20



Description

For constants, if the visibility is set to `public`, the compiler will automatically generate a getter function for it, which will consume more gas during deployment. The value of constant variable can be directly read from verified source code.

/hebao_v3/contracts/base/ConnectorRegistry.sol

```
9  contract ConnectorRegistry is AccessControl, Ownable {  
10    bytes32 public constant MANAGER = keccak256("MANAGER");  
11  
12    mapping(address => bool) public connectors;
```

/hebao_v3/contracts/base/LoopringPaymaster.sol

```
32    uint256 private constant COST_OF_POST = 60000;  
33    uint8 private constant PRICE_DECIMAL = 8;  
34    bytes32 public constant SIGNER = keccak256("SIGNER");  
35  
36    mapping(address => bool) public registeredToken;
```

/hebao_v3/contracts/base/WalletDeploymentLib.sol

```
13    address public immutable walletImplementation;  
14  
15    string public constant WALLET_CREATION = "WALLET_CREATION";  
16  
17    constructor(address _walletImplementation) {
```

/hebao_v3/contracts/base/WalletFactory.sol

```
28     bytes32 public constant CREATE_WALLET_TYPEHASH =
29         keccak256(
30             "createWallet(address owner,address[] guardians,uint256
31             quota,address inheritor,address feeRecipient,address
32             feeToken,uint256 maxFeeAmount,uint256 salt)"
```

/hebao_v3/contracts/base/libwallet/ERC20Lib.sol

```
34     bytes32 public constant TRANSFER_TOKEN_TYPEHASH =
35         keccak256(
36             "transferToken(address wallet,uint256 validUntil,address
37             token,address to,uint256 amount,bytes logdata)"
38         );
39     bytes32 public constant APPROVE_TOKEN_TYPEHASH =
40         keccak256(
41             "approveToken(address wallet,uint256 validUntil,address
42             token,address to,uint256 amount)"
43         );
44     bytes32 public constant CALL_CONTRACT_TYPEHASH =
45         keccak256(
46             "callContract(address wallet,uint256 validUntil,address to,
47             uint256 value,bytes data)"
48         );
49     bytes32 public constant APPROVE_THEN_CALL_CONTRACT_TYPEHASH =
50         keccak256(
51             "approveThenCallContract(address wallet,uint256 validUntil,
52             address token,address to,uint256 amount,uint256 value,bytes
53             data)"
54     );
```

/hebao_v3/contracts/base/libwallet/GuardianLib.sol

```
21     uint public constant MAX_GUARDIANS = 10;
22     uint public constant GUARDIAN_PENDING_PERIOD = 3 days;
23     SigRequirement public constant SIG_REQUIREMENT =
24         SigRequirement.MAJORITY_OWNER_REQUIRED;
25
26     bytes32 public constant ADD_GUARDIAN_TYPEHASH =
27         keccak256(
28             "addGuardian(address wallet,uint256 validUntil,address
29             guardian)"
30         );
31     bytes32 public constant REMOVE_GUARDIAN_TYPEHASH =
32         keccak256(
33             "removeGuardian(address wallet,uint256 validUntil,address
34             guardian)"
35         );
36     bytes32 public constant RESET_GUARDIANS_TYPEHASH =
37         keccak256(
38             "resetGuardians(address wallet,uint256 validUntil,address[]
39             guardians)"
40         );
```

/hebao_v3/contracts/base/libwallet/InheritanceLib.sol

```
19     // The minimal number of guardians for recovery and locking.
20     uint public constant TOUCH_GRACE_PERIOD = 30 days;
21
22     event Inherited(address inheritor, address newOwner);
```

/hebao_v3/contracts/base/libwallet/LockLib.sol

```
22     bytes32 public constant UNLOCK_TYPEHASH =
23         keccak256("unlock(address wallet,uint256 validUntil)");
24
```

/hebao_v3/contracts/base/libwallet/QuotaLib.sol

```
21     uint128 public constant MAX_QUOTA = type(uint128).max;
22     uint public constant QUOTA_PENDING_PERIOD = 1 days;
23     SigRequirement public constant SIG_REQUIREMENT =
24         SigRequirement.MAJORITY_OWNER_REQUIRED;
25
26     bytes32 public constant CHANGE_DAILY_QUOTE_TYPEHASH =
27         keccak256(
28             "changeDailyQuota(address wallet,uint256 validUntil,uint256
29             newQuota)"
29     );
```

/hebao_v3/contracts/base/libwallet/RecoverLib.sol

```
24     bytes32 public constant RECOVER_TYPEHASH =
25         keccak256(
26             "recover(address wallet,uint256 validUntil,address newOwner,
27             address[] newGuardians)"
```

/hebao_v3/contracts/base/libwallet/UpgradeLib.sol

```
20     bytes32 public constant CHANGE_MASTER_COPY_TYPEHASH =
21         keccak256(
22             "changeMasterCopy(address wallet,uint256 validUntil,address
23             masterCopy)"
```

/hebao_v3/contracts/base/libwallet/WhitelistLib.sol

```
16     uint public constant WHITELIST_PENDING_PERIOD = 1 days;
17     SigRequirement public constant SIG_REQUIREMENT =
18         SigRequirement.MAJORITY_OWNER_REQUIRED;
19
20     bytes32 public constant ADD_TO_WHITELIST_TYPEHASH =
21         keccak256(
22             "addToWhitelist(address wallet,uint256 validUntil,address
23             addr)"
24         );

```

/hebao_v3/contracts/connectors/base_connector.sol

```
22
23 contract DSMath {
24     uint public constant WAD = 10 ** 18;
25     uint public constant RAY = 10 ** 27;
26

```

/hebao_v3/contracts/connectors/memory.sol

```
76 contract OwnedMemory is Memory {
77     address public master;
78     address public constant BROADCAST_ADDR = address(0);

```

/hebao_v3/contracts/price/CachedPriceOracle.sol

```
15     uint public constant EXPIRY_PERIOD = 7 days;
```

/hebao_v3/contracts/price/KyberNetworkPriceOracle.sol

```
18 contract KyberNetworkPriceOracle is PriceOracle {
19     KyberNetworkProxy public immutable kyber;
20     address public constant ETH_ADDR =
21         0xEeeeeEeeeEeEeEeeEEEeeeeEeeeeeeeEEeE;
```

Recommendation

It is recommended to set the visibility of constants to `private` instead of `public`.

Alleviation

Acknowledged and partially applied in commit 49167c6.

I-18: Interface Defined Not Inherited



Informational: Optimization Suggestion

File Location: /hebao_v3/contracts/base/ConnectorRegistry.sol:9

Description

Interface `IConnectorRegistry` is defined but not properly inherited by contract `ConnectorRegistry`.

/hebao_v3/contracts/base/ConnectorRegistry.sol

```
7 import "@openzeppelin/contracts/access/AccessControl.sol";
8
9 contract ConnectorRegistry is AccessControl, Ownable {
10     bytes32 public constant MANAGER = keccak256("MANAGER");
11 }
```

Recommendation

It is recommended that the contract inherits the relevant interface when it is defined.

Alleviation

Resolved in commit 49167c6. `ConnectorRegistry` now inherits `IConnectorRegistry`.

4. Disclaimer

No description, statement, recommendation or conclusion in this report shall be construed as endorsement, affirmation or confirmation of the project. The security assessment is limited to the scope of work as stipulated in the Statement of Work.

This report is prepared in response to source code, and based on the attacks and vulnerabilities in the source code that already existed or occurred before the date of this report, excluding any new attacks or vulnerabilities that exist or occur after the date of this report. The security assessment are solely based on the documents and materials provided by the customer, and the customer represents and warrants documents and materials are true, accurate and complete.

CONSULTANT DOES NOT MAKE AND HEREBY DISCLAIMS ANY REPRESENTATIONS OR WARRANTIES OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, REGARDING THE SERVICES, DELIVERABLES, OR ANY OTHER MATTER PERTAINING TO THIS REPORT.

CONSULTANT SHALL NOT BE RESPONSIBLE FOR AND HEREBY DISCLAIMS MERCHANTABILITY, FITNESS FOR PURPOSE, TITLE, NON-INFRINGEMENT OR NON-APPROPRIATION OF INTELLECTUAL PROPERTY RIGHTS OF A THIRD PARTY, SATISFACTORY QUALITY, ACCURACY, QUALITY, COMPLETENESS, TIMELINESS, RESPONSIVENESS, OR PRODUCTIVITY OF THE SERVICES OR DELIVERABLES.

CONSULTANT EXCLUDES ANY WARRANTY THAT THE SERVICES AND DELIVERABLES WILL BE UNINTERRUPTED, ERROR FREE, FREE OF SECURITY DEFECTS OR HARMFUL COMPONENTS, REVEAL ALL SECURITY VULNERABILITIES, OR THAT ANY DATA WILL NOT BE LOST OR CORRUPTED.

CONSULTANT SHALL NOT BE RESPONSIBLE FOR (A) ANY REPRESENTATIONS MADE BY ANY PERSON REGARDING THE SUFFICIENCY OR SUITABILITY OF SERVICES AND DELIVERABLES IN ANY ACTUAL APPLICATION, OR (B) WHETHER ANY SUCH USE WOULD VIOLATE OR INFRINGE THE APPLICABLE LAWS, OR (C) REVIEWING THE CUSTOMER MATERIALS FOR ACCURACY.

5. Appendix

5.1 Visibility

| Contract | FuncName | Visibility | Mutability | Modifiers |
|-------------------|------------------|------------|------------|-------------|
| BaseConnector | _CTOR_ | public | | |
| BaseConnector | getUint | internal | | |
| BaseConnector | setUint | internal | | |
| BaseConnector | changeEthAddress | internal | | |
| BaseConnector | convertEthToWeth | internal | | |
| BaseConnector | convertWethToEth | internal | | |
| BaseConnector | convert18ToDec | internal | | |
| BaseConnector | convertTo18 | internal | | |
| BaseConnector | getTokenBal | internal | | |
| LidoConnector | _CTOR_ | public | | |
| LidoConnector | deposit | public | | |
| LidoConnector | wrapAndStaking | public | | |
| LidoConnector | unwrap | public | | |
| WETHConnector | _CTOR_ | public | | |
| WETHConnector | deposit | public | | |
| WETHConnector | withdraw | public | | |
| ConnectorRegistry | _CTOR_ | public | | |
| ConnectorRegistry | addConnectors | external | | onlyManager |
| ConnectorRegistry | removeConnectors | external | | onlyManager |

| Contract | FuncName | Visibility | Mutability | Modifiers |
|---------------------|------------------------|-------------------|-------------------|------------------|
| ConnectorRegistry | isConnectors | external | | |
| WalletDeploymentLib | _CTOR_ | public | | |
| WalletDeploymentLib | getWalletCode | public | | |
| WalletDeploymentLib | computeWalletSalt | public | | |
| WalletDeploymentLib | _deploy | internal | | |
| WalletDeploymentLib | _computeWalletAddress | internal | | |
| WalletFactory | isOperator | public | | |
| WalletFactory | operators | public | | |
| WalletFactory | numOperators | public | | |
| WalletFactory | addOperator | public | | onlyOwner |
| WalletFactory | removeOperator | public | | onlyOwner |
| WalletFactory | addOperatorInternal | internal | | |
| WalletFactory | _CTOR_ | public | | |
| WalletFactory | createWallet | external | | onlyOperator |
| WalletFactory | createWalletByOperator | external | | onlyOperator |
| WalletFactory | computeWalletAddress | public | | |
| WalletFactory | _initializeWallet | internal | | |
| WalletFactory | _validateConfig | private | | |

| Contract | FuncName | Visibility | Mutability | Modifiers |
|------------------------------|------------------|-------------------|-------------------|------------------|
| DelayedImplementationManager | _CTOR_ | public | | |
| DelayedImplementationManager | delayedUpgradeTo | public | | onlyOwner |
| DelayedImplementationManager | executeUpgrade | public | | |
| ForwardProxy | _CTOR_ | public | | |
| ForwardProxy | _implementation | internal | | |
| KyberNetworkPriceOracle | _CTOR_ | public | | |
| KyberNetworkPriceOracle | tokenValue | public | | |
| OwnedMemory | _CTOR_ | public | | |
| OwnedMemory | getBroadcastAddr | public | | |
| OwnedMemory | setBroadcastAddr | public | | isMaster |
| CachedPriceOracle | _CTOR_ | public | | |
| CachedPriceOracle | tokenValue | public | | |
| CachedPriceOracle | updateTokenPrice | external | | onlyManager |
| CachedPriceOracle | setTokenPrice | external | | onlyManager |
| CachedPriceOracle | setOracle | external | | onlyManager |
| CachedPriceOracle | _cacheTokenPrice | internal | | |
| LoopringPaymaster | _CTOR_ | public | | |
| LoopringPaymaster | receive | external | | |
| LoopringPaymaster | getHash | public | | |

| Contract | FuncName | Visibility | Mutability | Modifiers |
|--------------------|--------------------------|-------------------|-------------------|------------------|
| LoopringPaymaster | _validatePaymasterUserOp | internal | | |
| LoopringPaymaster | parsePaymasterAndData | public | | |
| LoopringPaymaster | _postOp | internal | | |
| LoopringPaymaster | addToken | external | | onlyOwner |
| LoopringPaymaster | removeToken | external | | onlyOwner |
| LoopringPaymaster | addDepositFor | external | | |
| LoopringPaymaster | depositInfo | public | | |
| LoopringPaymaster | unlockTokenDeposit | public | | |
| LoopringPaymaster | lockTokenDeposit | public | | |
| LoopringPaymaster | withdrawTokensTo | public | | |
| BalancerFlashLoan | _CTOR_ | public | | |
| BalancerFlashLoan | receiveFlashLoan | external | | |
| BalancerFlashLoan | flashLoan | external | | |
| FlashLoanConnector | _CTOR_ | public | | |
| FlashLoanConnector | flashBorrowAndCast | external | | |
| FlashLoanConnector | flashPayback | external | | |
| AaveV3Connector | _CTOR_ | public | | |
| AaveV3Connector | getIsColl | internal | | |
| AaveV3Connector | getPaybackBalance | internal | | |

| Contract | FuncName | Visibility | Mutability | Modifiers |
|---------------------------|--------------------|-------------------|-------------------|------------------|
| AaveV3Connector | deposit | external | | |
| AaveV3Connector | withdraw | external | | |
| AaveV3Connector | borrow | external | | |
| AaveV3Connector | payback | external | | |
| UniswapConnector | _CTOR_ | public | | |
| UniswapConnector | deposit | external | | |
| UniswapConnector | withdraw | external | | |
| UniswapConnector | buy | external | | |
| UniswapConnector | sell | external | | |
| UniswapConnector | getMinAmount | internal | | |
| UniswapConnector | _addLiquidity | internal | | |
| UniswapConnector | _removeLiquidity | internal | | |
| UniswapConnector | getExpectedBuyAmt | internal | | |
| UniswapConnector | getExpectedSellAmt | internal | | |
| UniswapConnector | checkPair | internal | | |
| UniswapConnector | getPaths | internal | | |
| AggregationalPrice Oracle | _CTOR_ | public | | |
| AggregationalPrice Oracle | tokenValue | public | | |
| CompoundConnector | _CTOR_ | public | | |
| CompoundConnector | deposit | public | | |

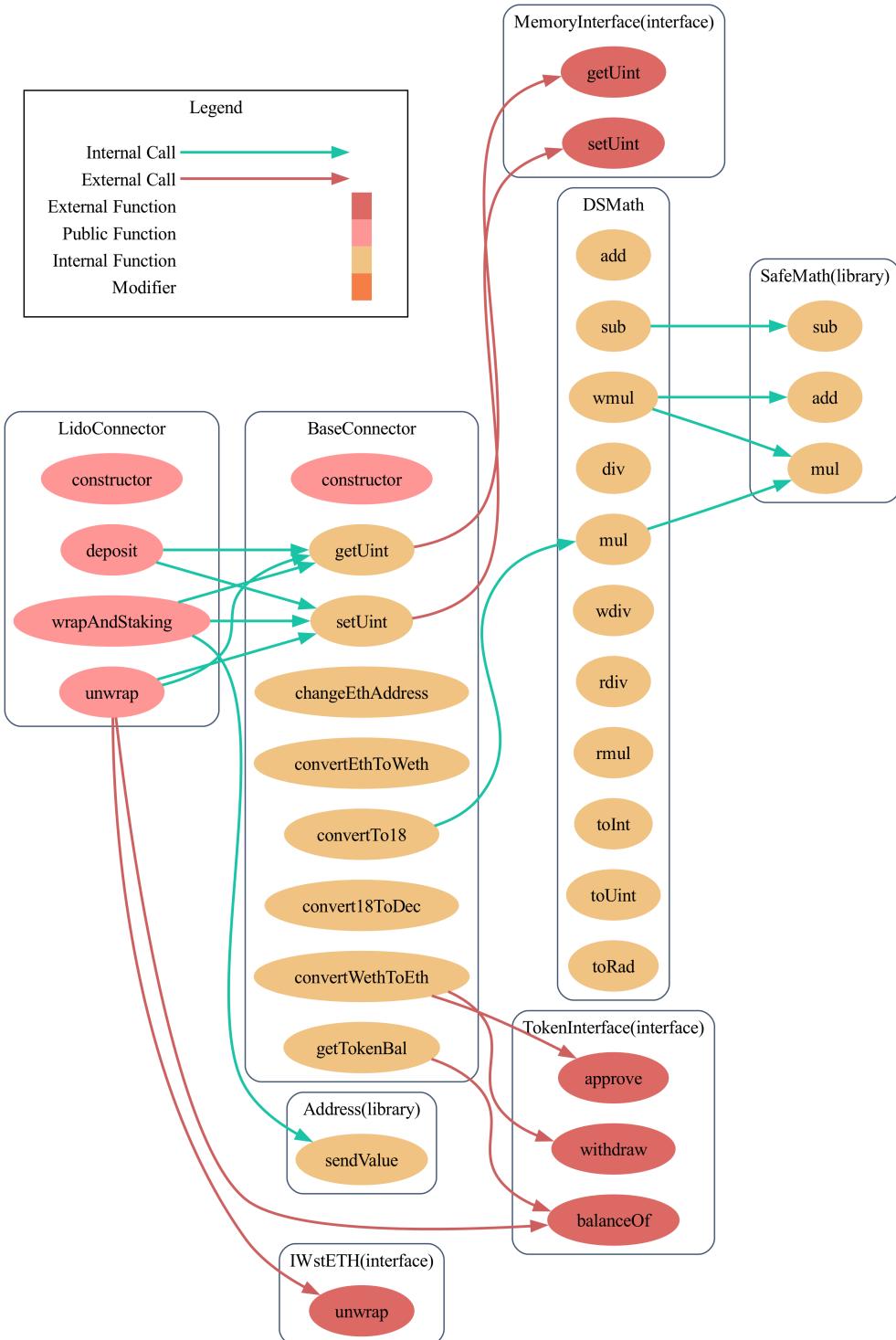
| Contract | FuncName | Visibility | Mutability | Modifiers |
|-------------------------|--------------------|-------------------|-------------------|---|
| CompoundConnect or | withdraw | public | | |
| CompoundConnect or | borrow | public | | |
| CompoundConnect or | payback | public | | |
| CompoundConnect or | depositCToken | public | | |
| CompoundConnect or | withdrawCToken | public | | |
| CompoundConnect or | liquidate | public | | |
| CompoundConnect or | enterMarket | internal | | |
| UniswapV2PriceOracle | _CTOR_ | public | | |
| UniswapV2PriceOracle | tokenValue | public | | |
| SmartWalletV3 | _CTOR_ | public | | |
| SmartWalletV3 | selfBatchCall | external | | onlyFromEntryPoint OrWalletOrOwnerWhenUnlocked |
| SmartWalletV3 | getDeposit | public | | |
| SmartWalletV3 | addDeposit | public | | |
| SmartWalletV3 | withdrawDepositTo | public | | onlyFromEntryPoint OrWalletOrOwnerWhenUnlocked |
| SmartWalletV3 | _validateSignature | internal | | |
| LoopringCreate2Deployer | deploy | public | | |
| OfficialGuardian | initOwner | external | | |

| Contract | FuncName | Visibility | Mutability | Modifiers |
|------------------|------------------|-------------------|-------------------|------------------|
| OfficialGuardian | isValidSignature | public | | |
| OfficialGuardian | transact | external | | onlyManager |

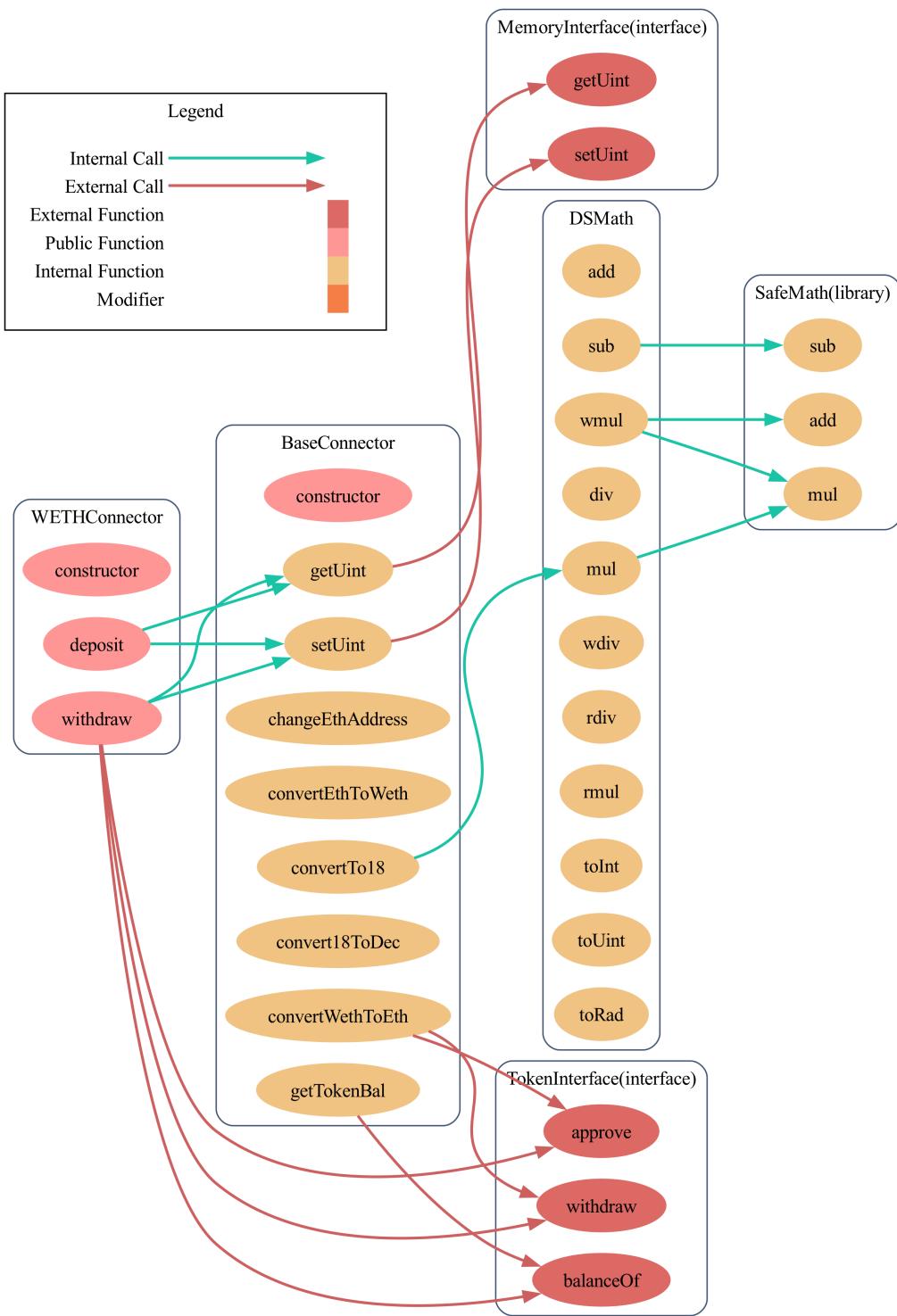
5. Appendix

5.2 Call Graph

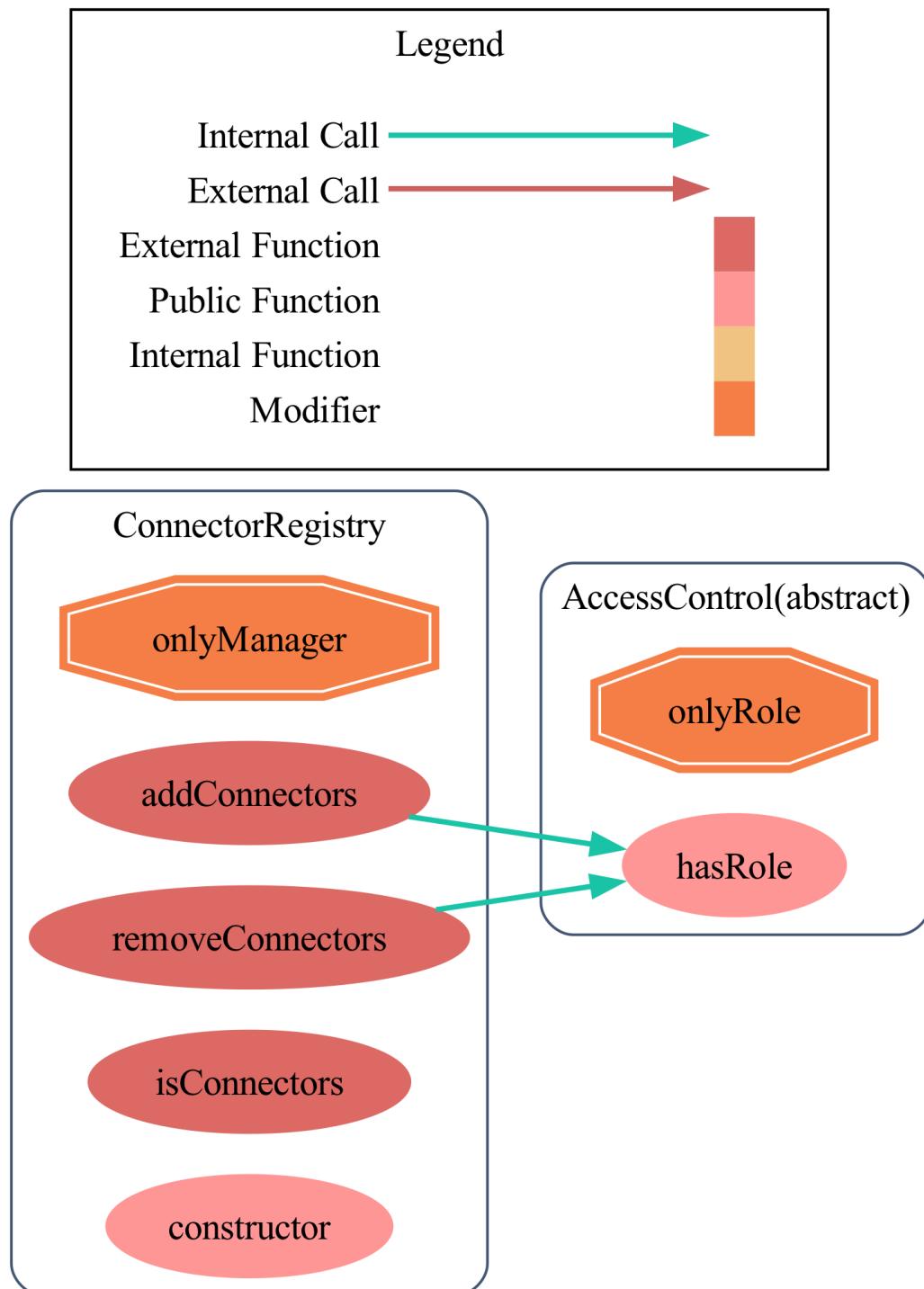
LidoConnector



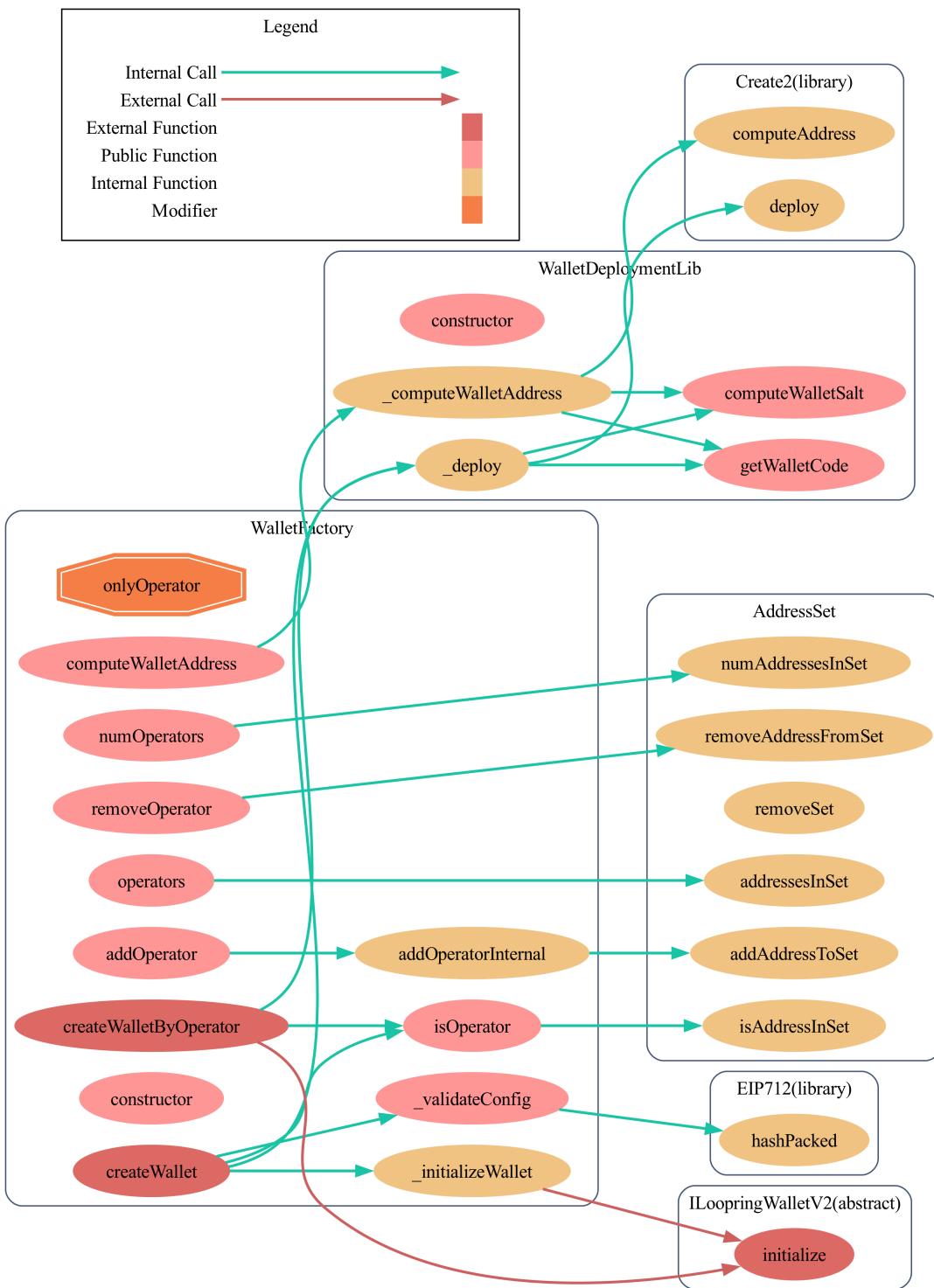
WETHConnector



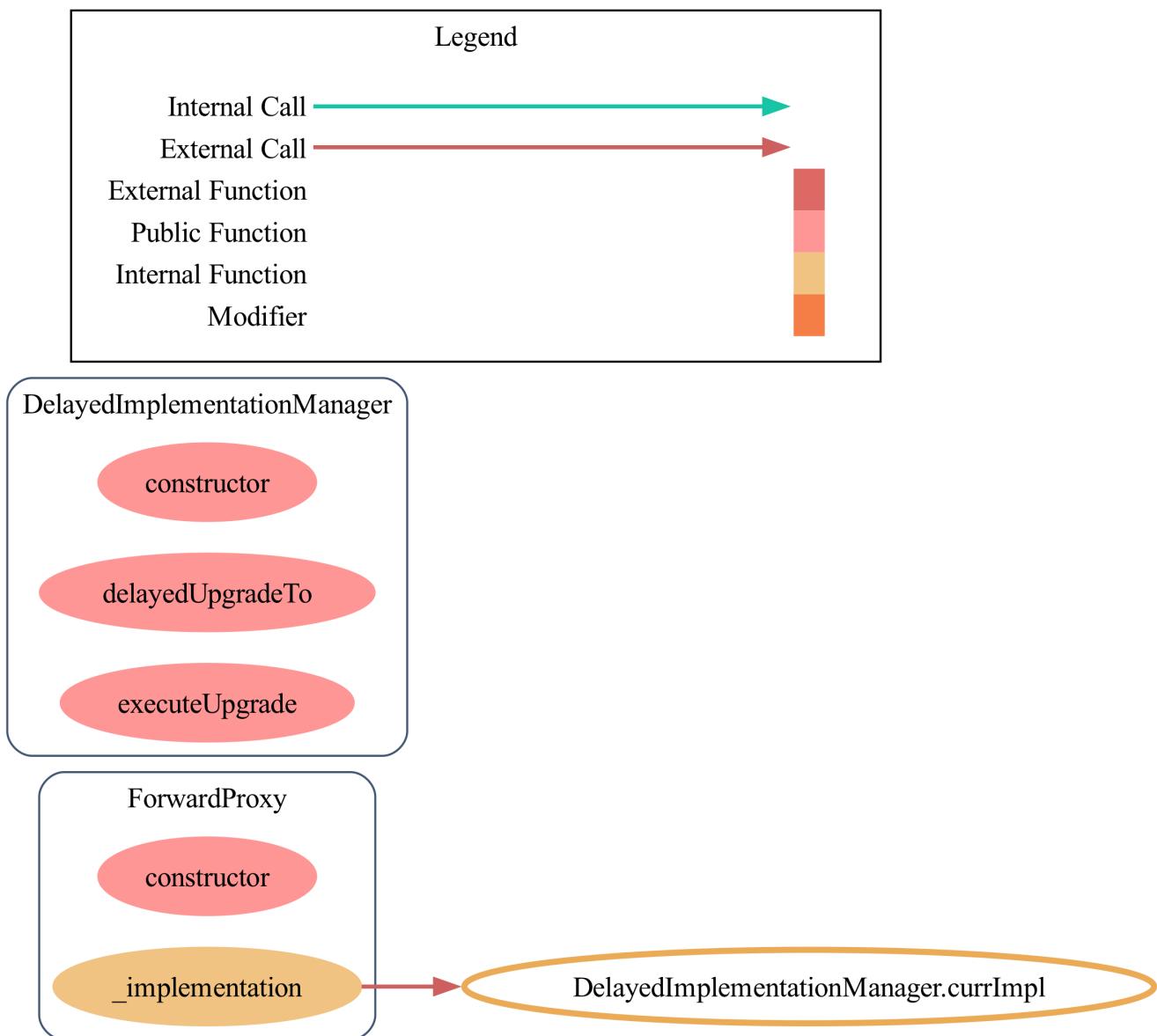
ConnectorRegistry



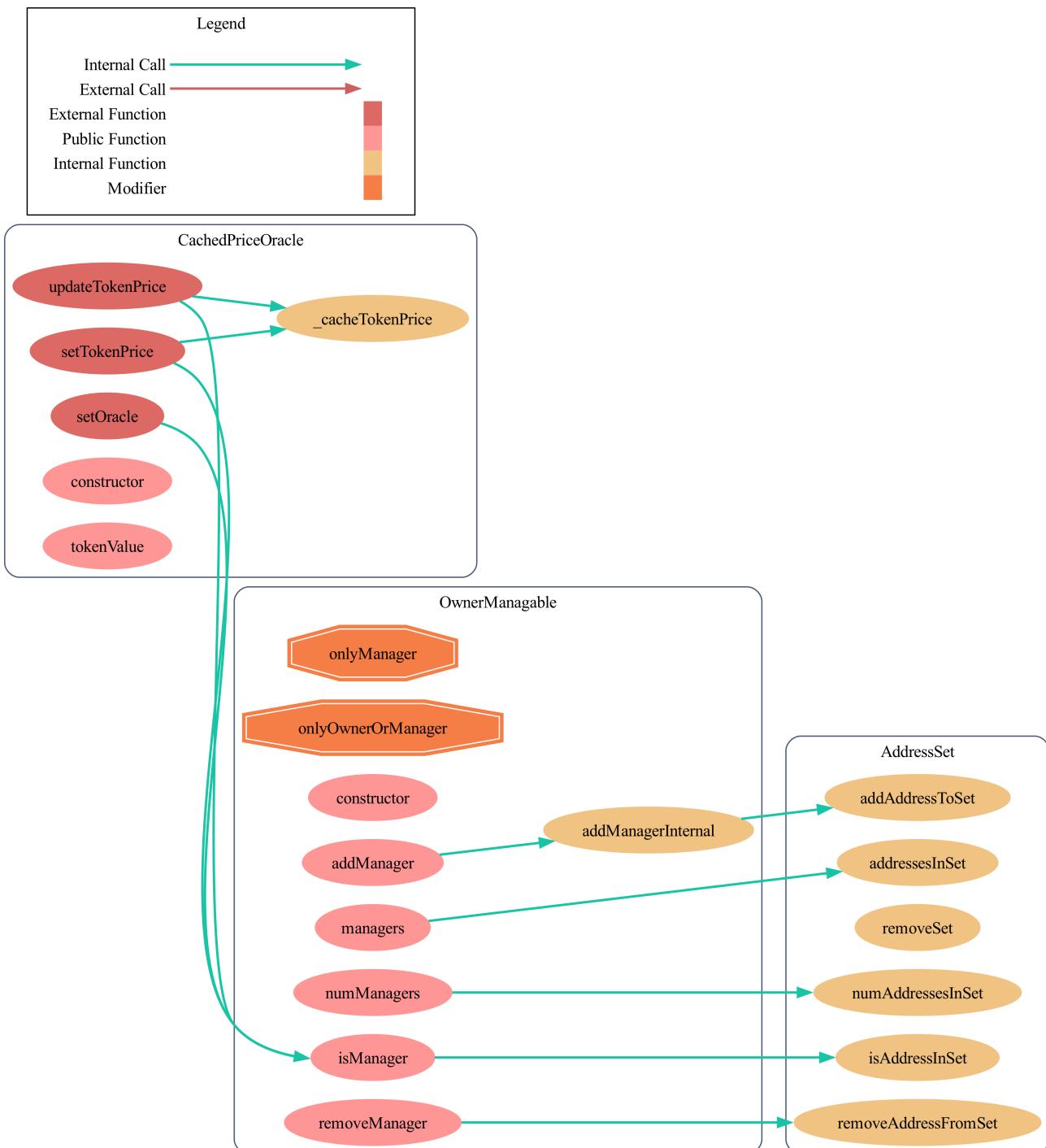
WalletFactory



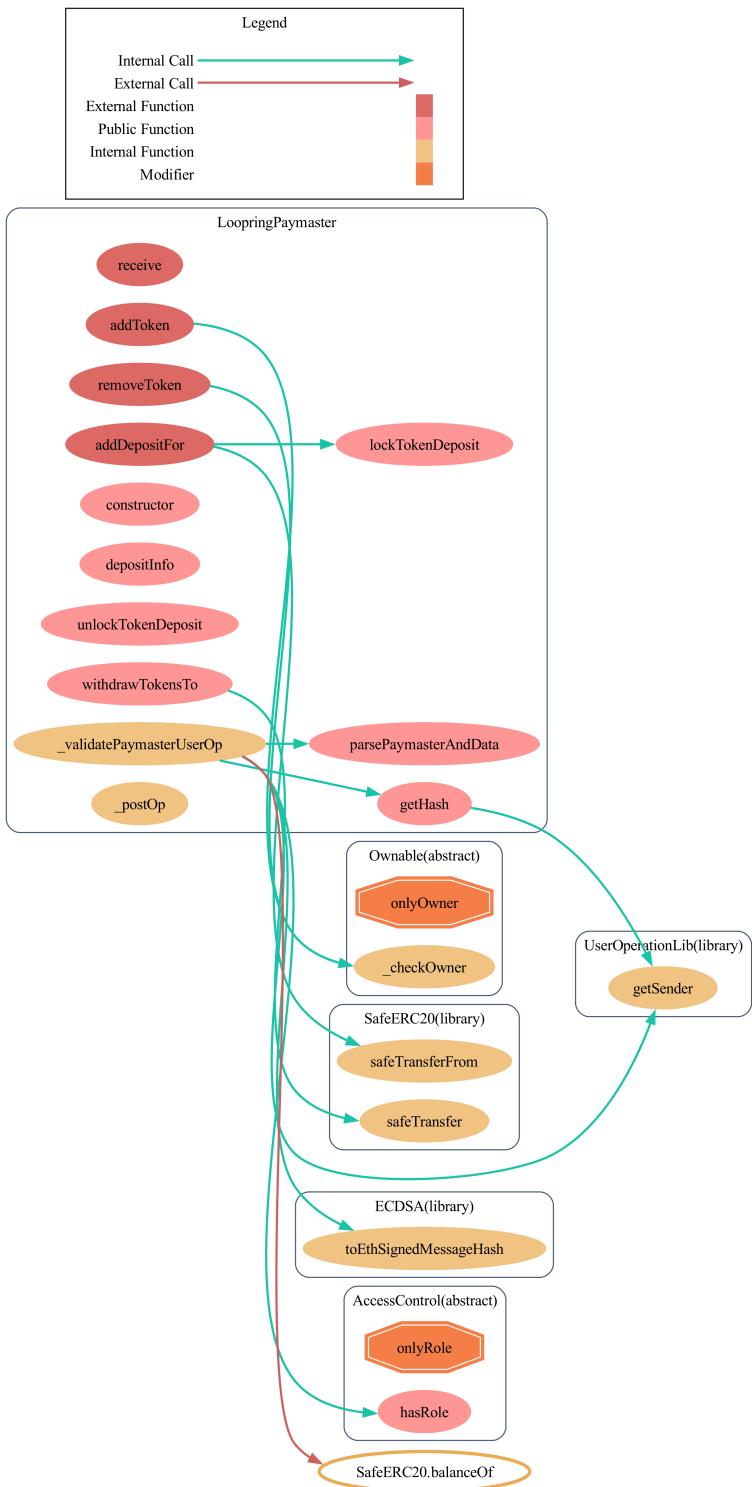
ForwardProxy



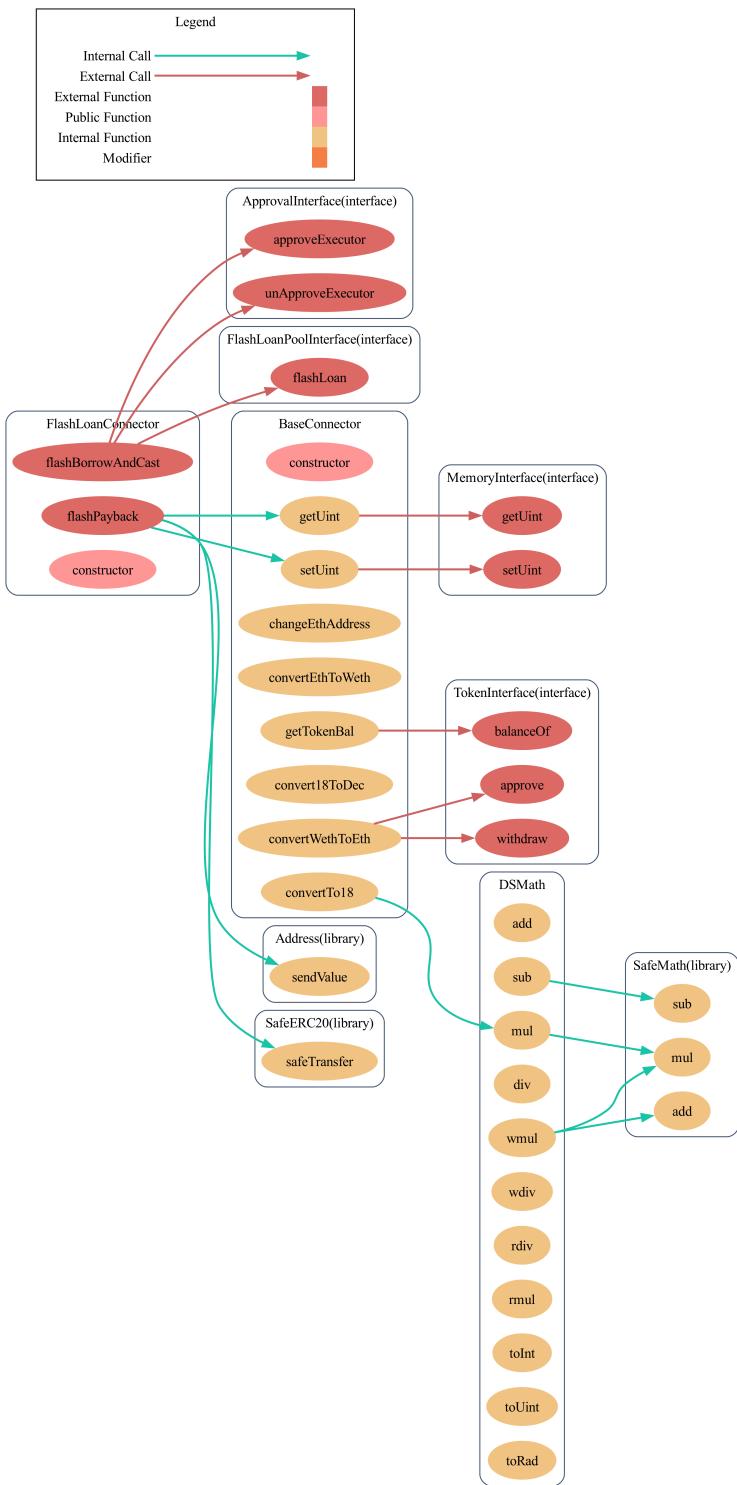
CachedPriceOracle



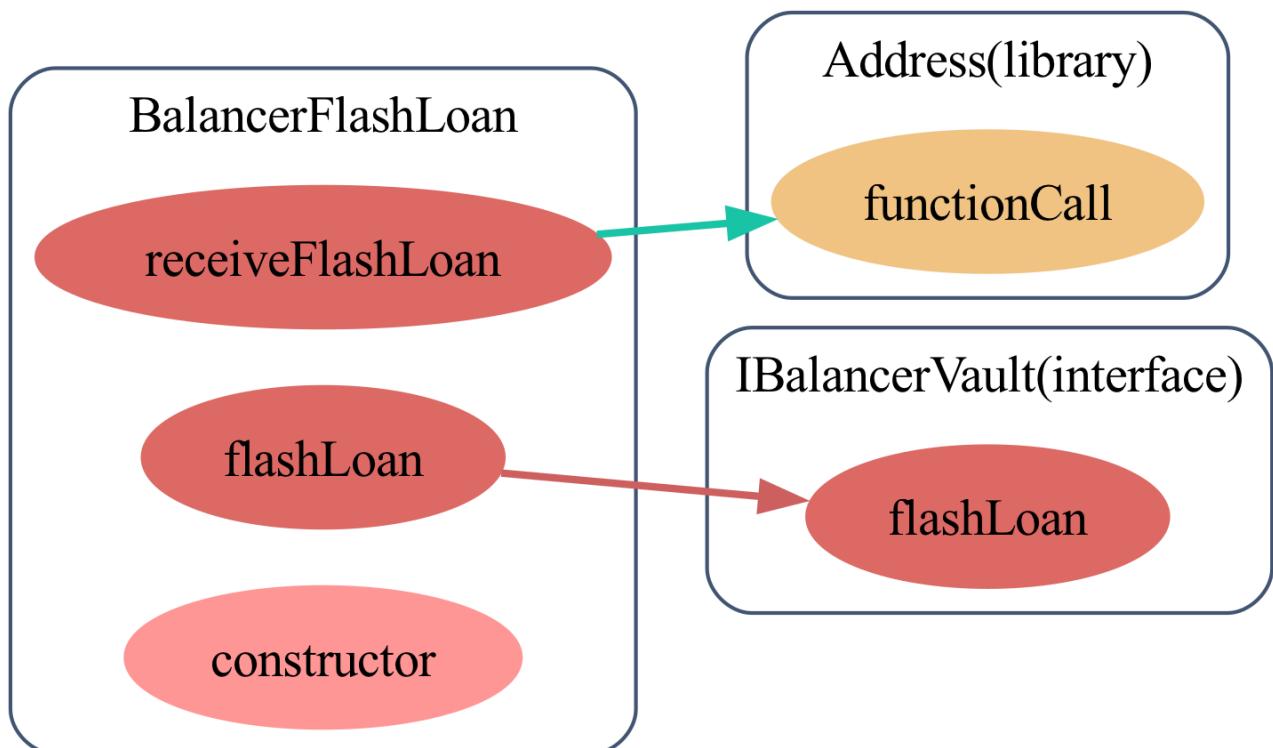
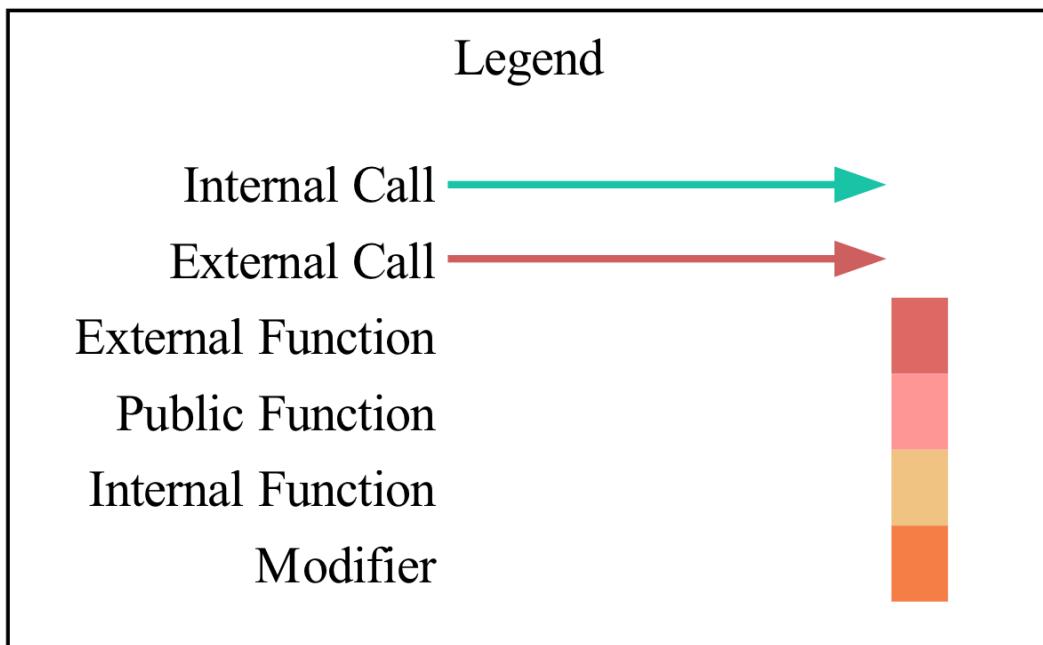
LoopringPaymaster



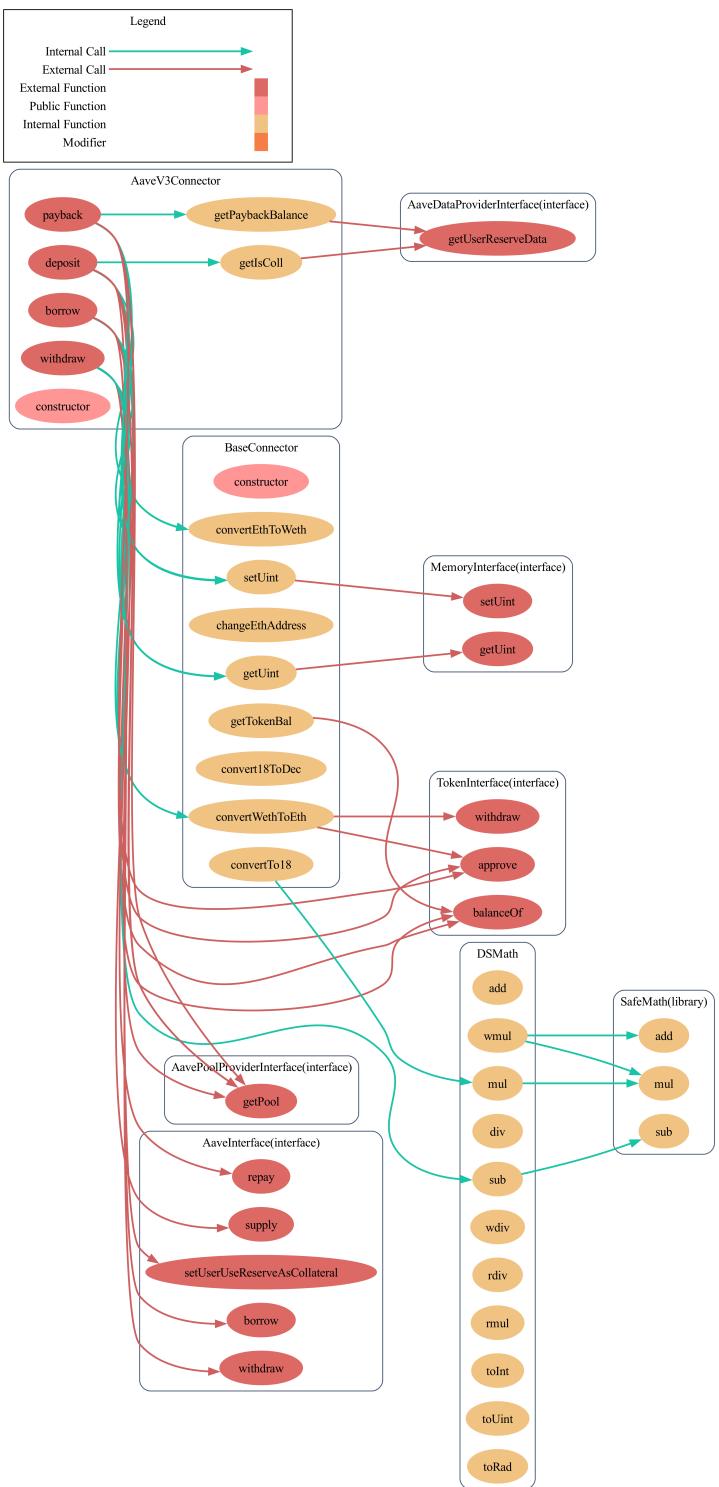
FlashLoanConnector



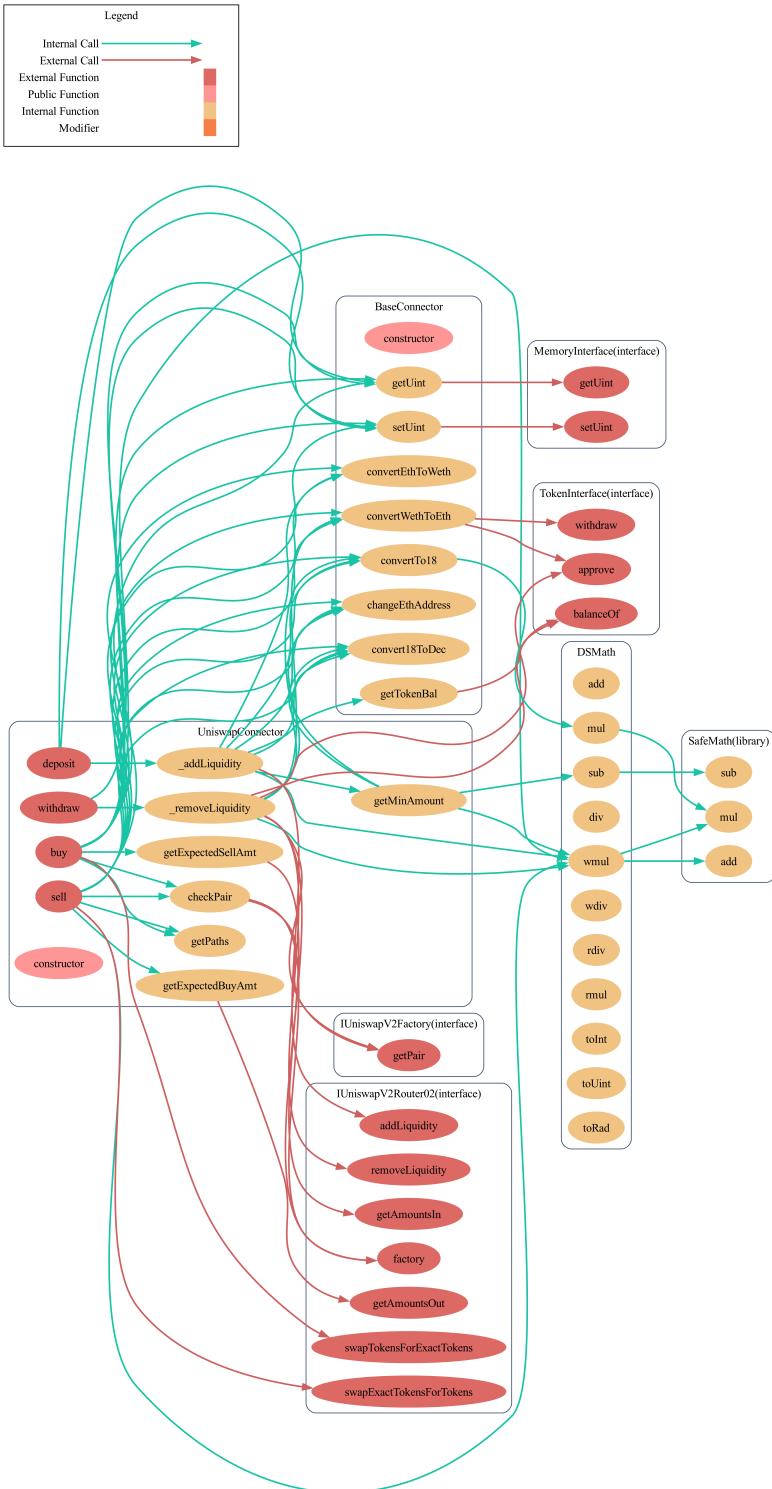
BalancerFlashLoan



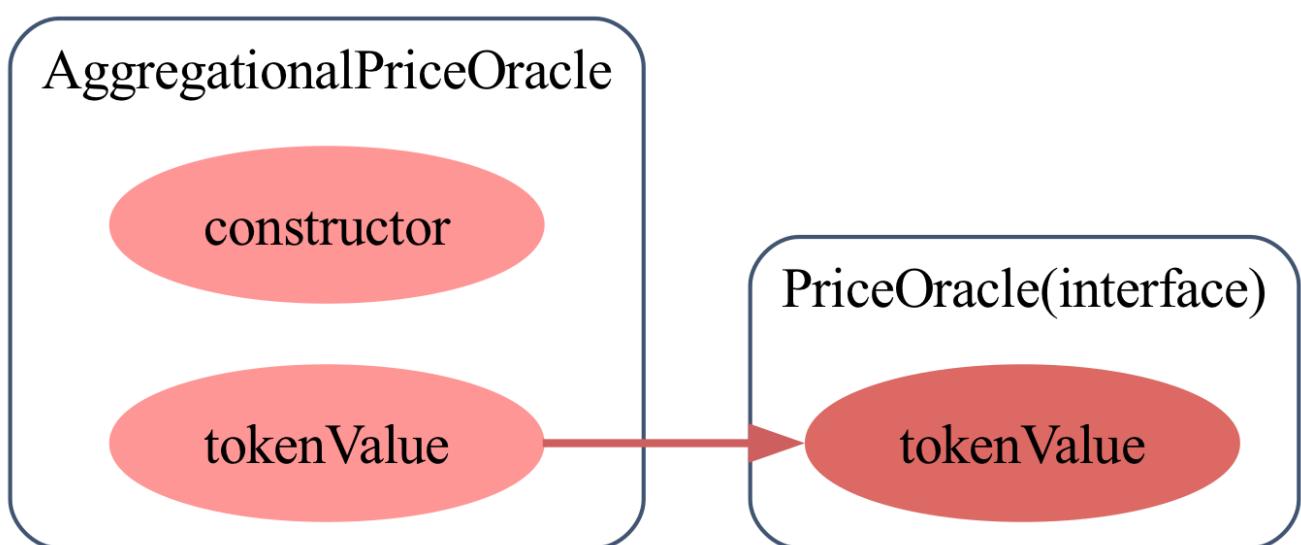
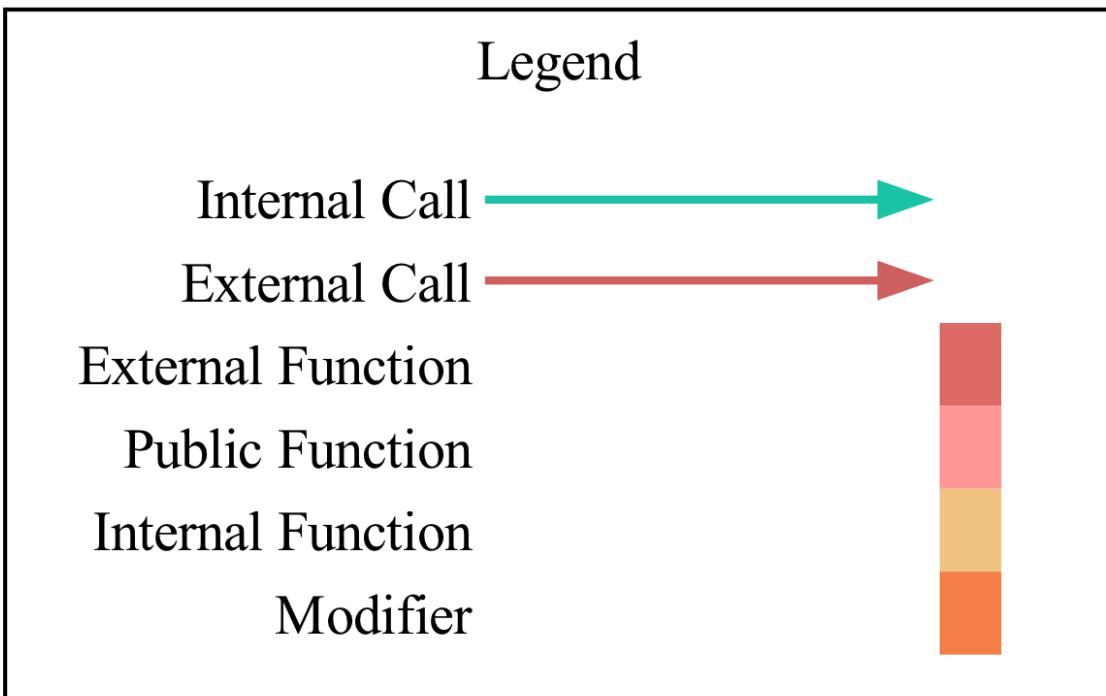
AaveV3Connector



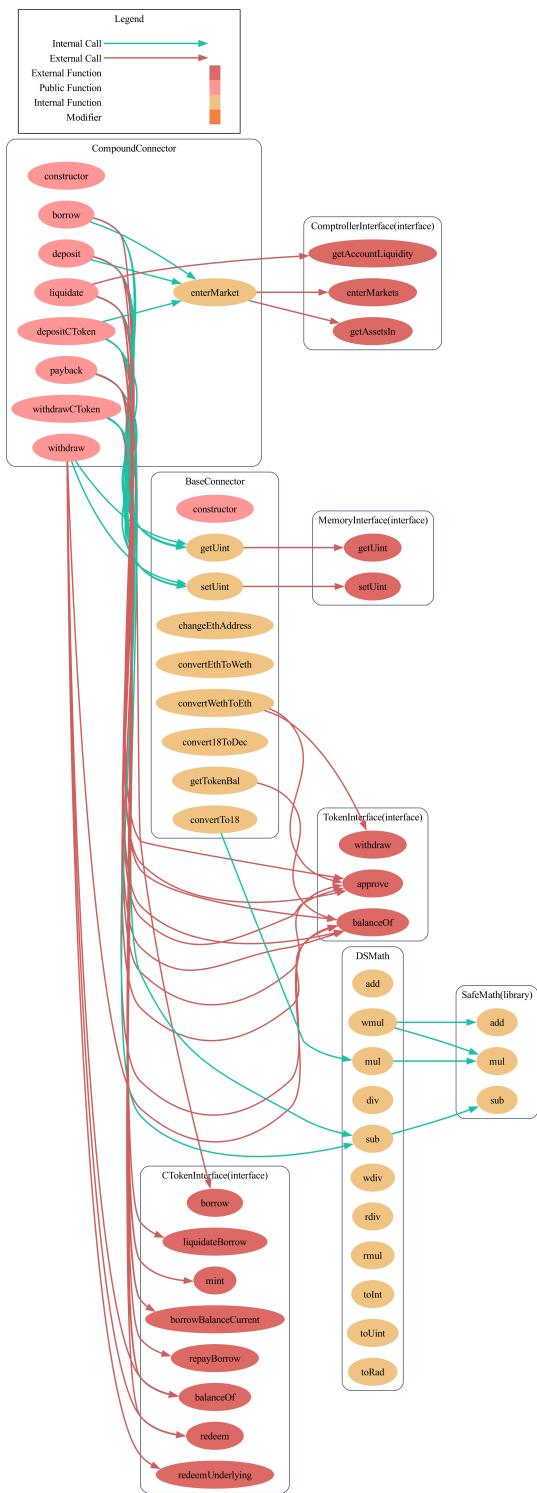
UniswapConnector



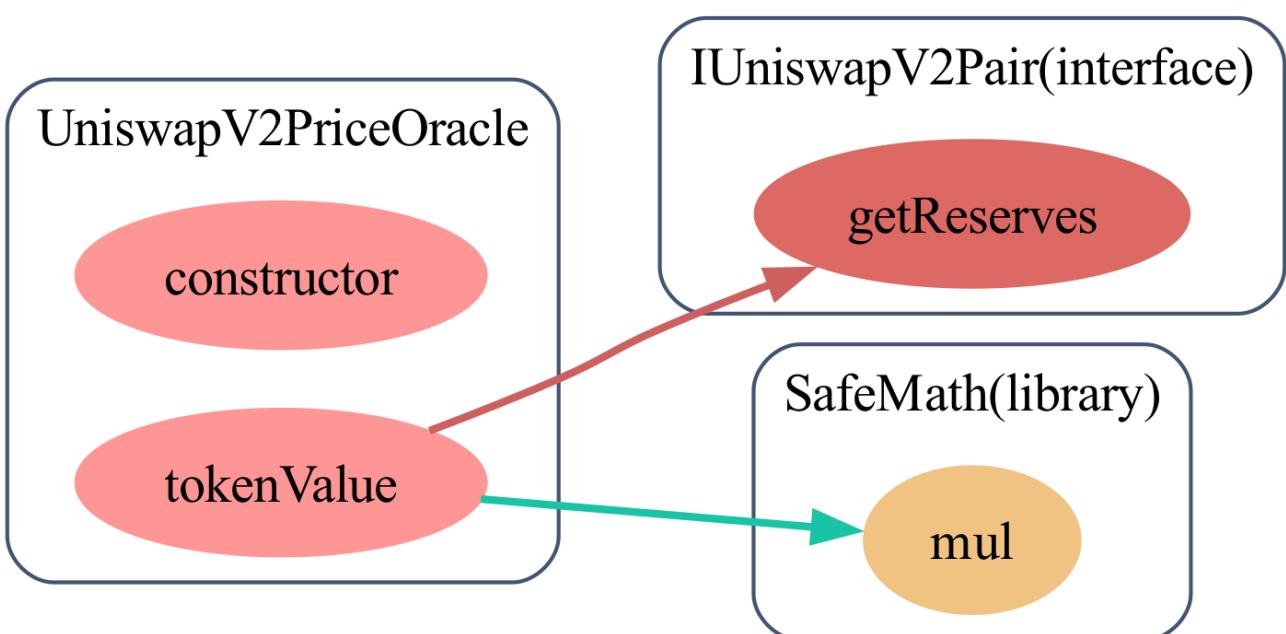
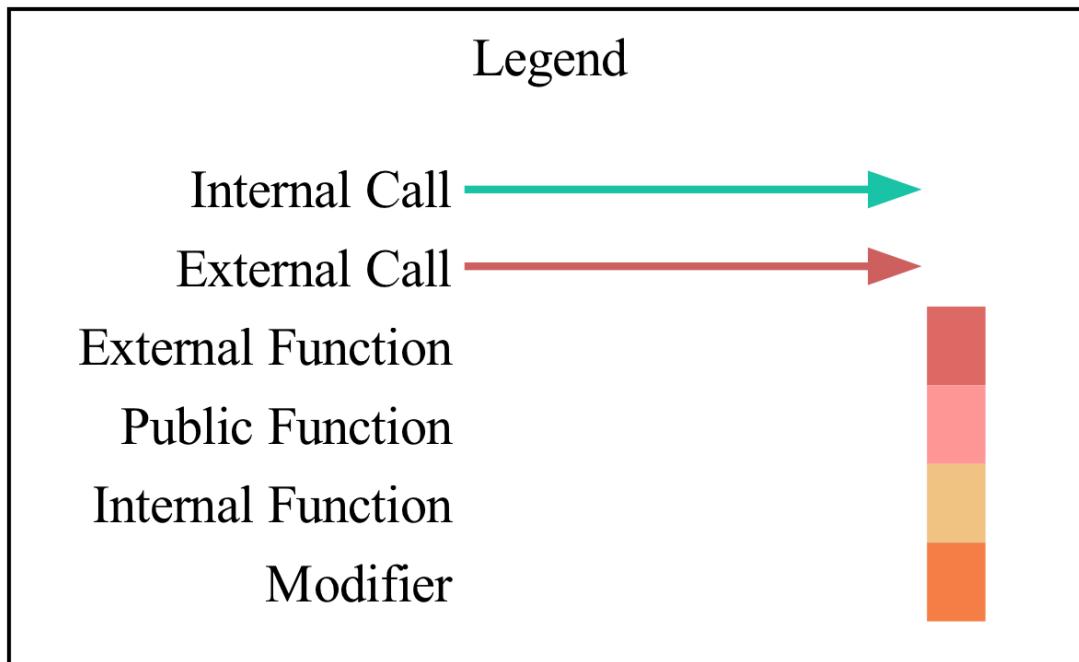
AggregationalPriceOracle



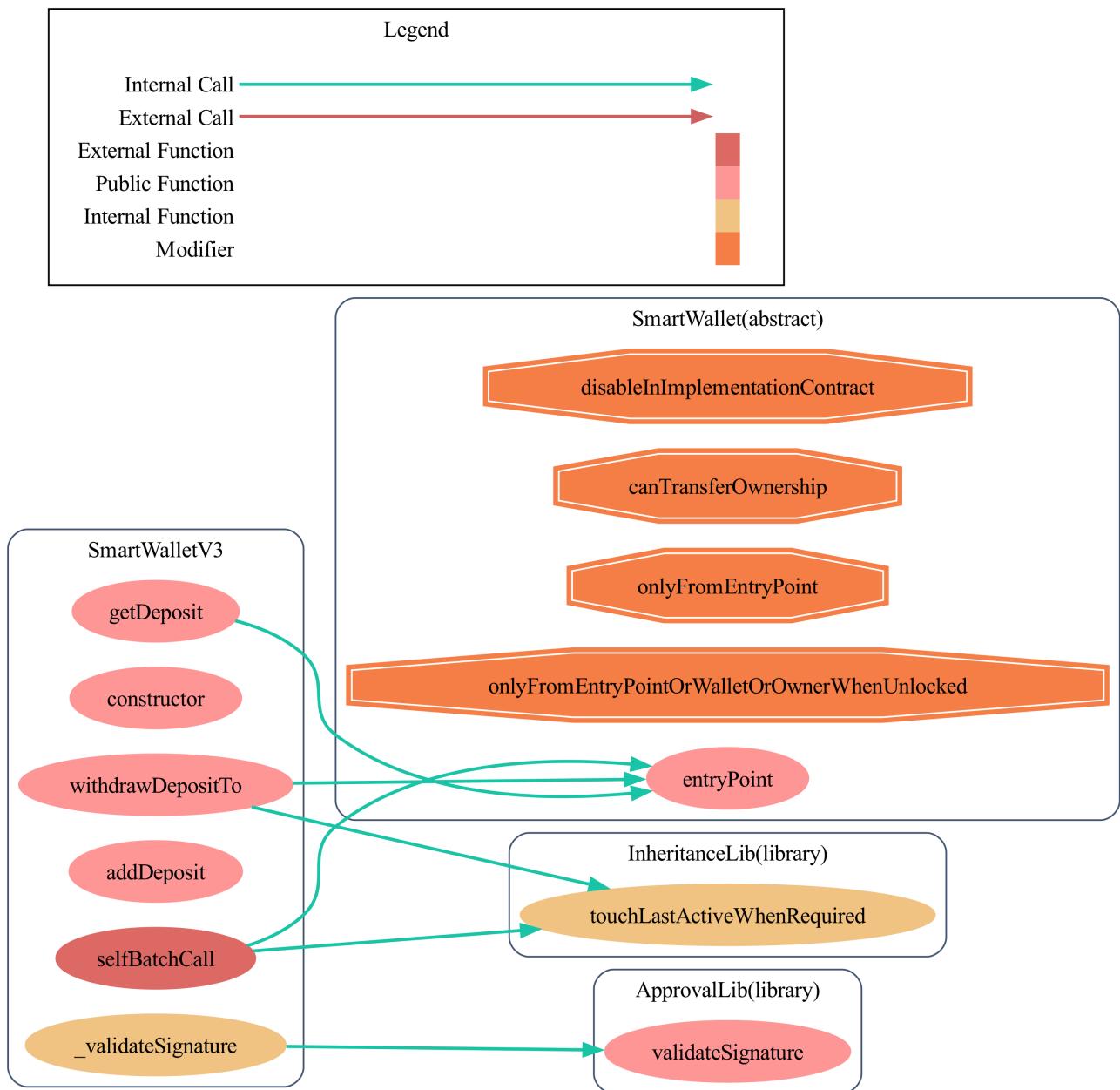
CompoundConnector



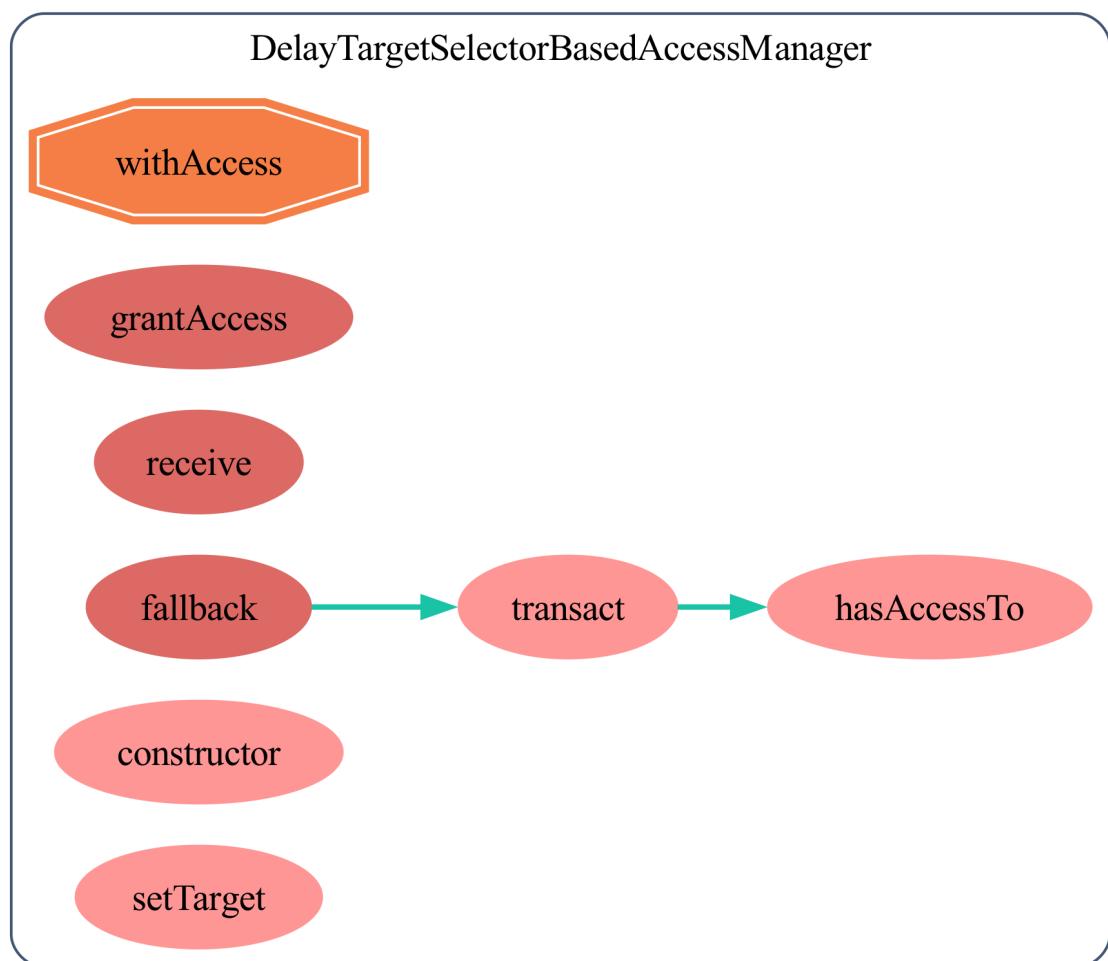
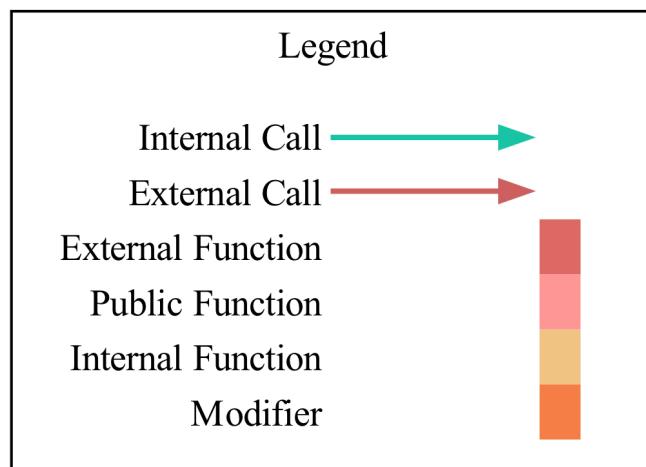
UniswapV2PriceOracle



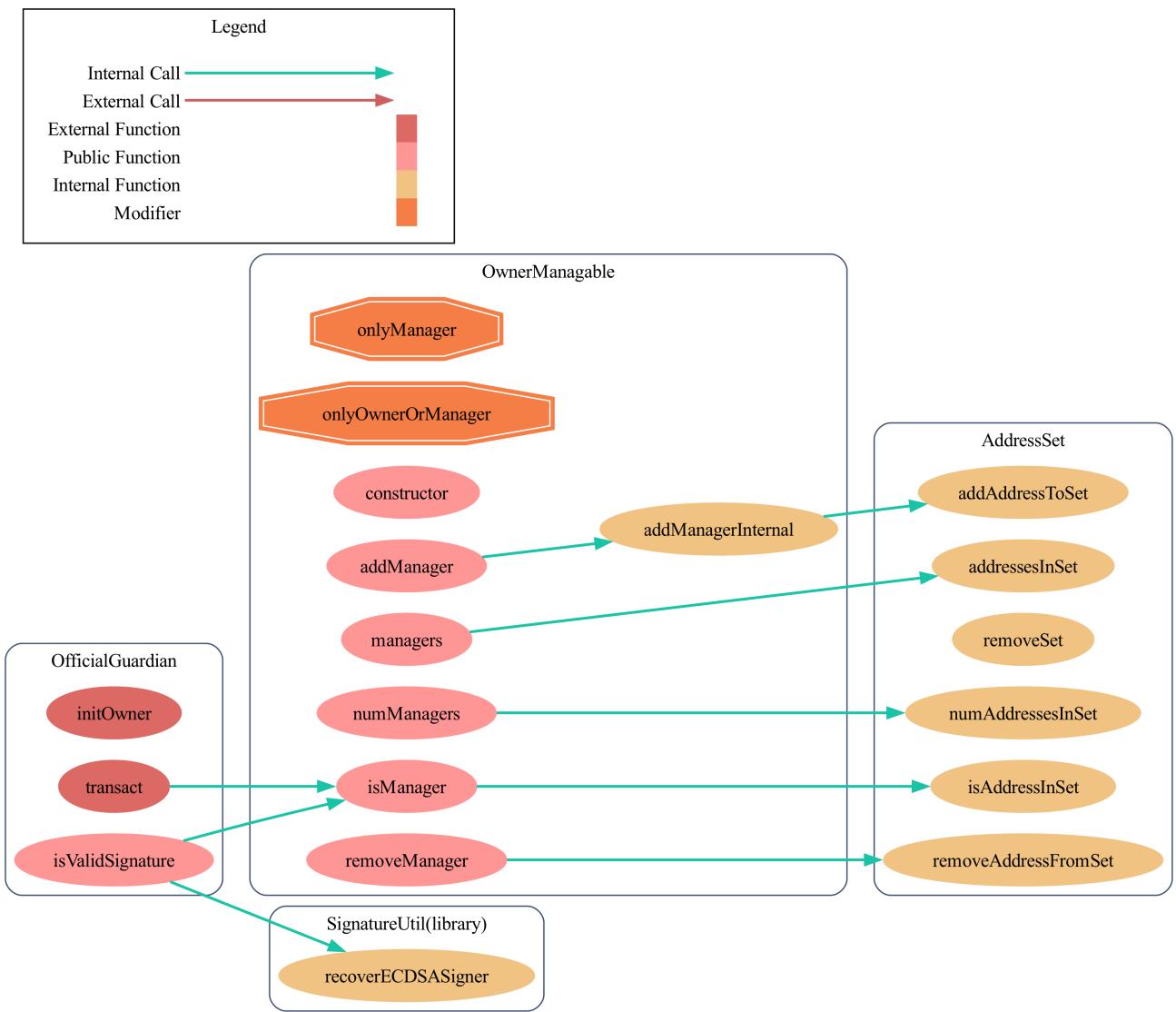
SmartWalletV3



LoopringCreate2Deployer



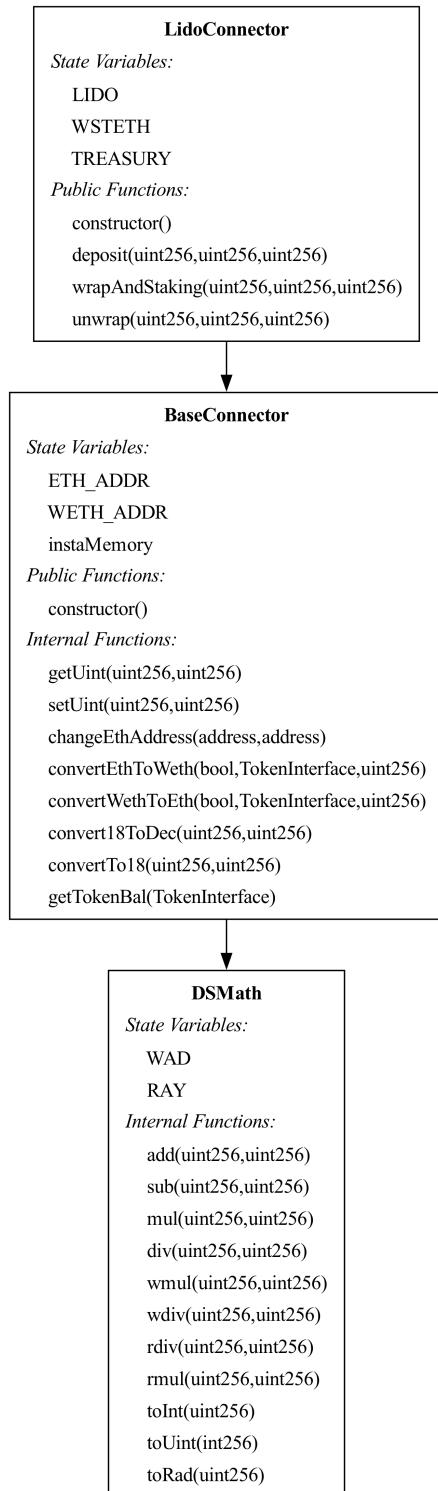
OfficialGuardian



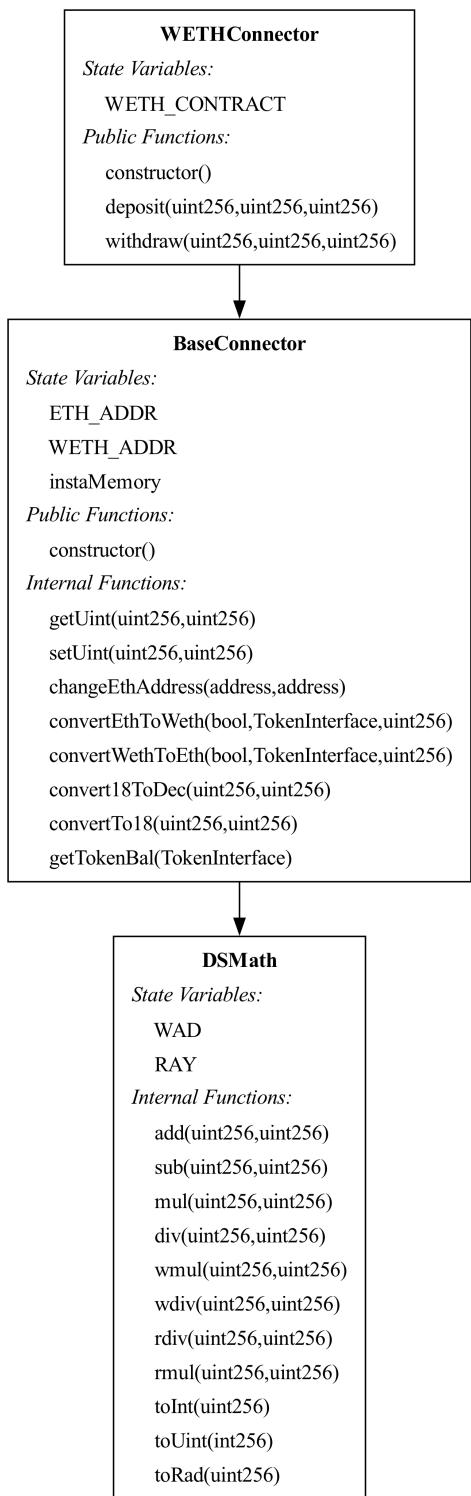
5. Appendix

5.3 Inheritance Graph

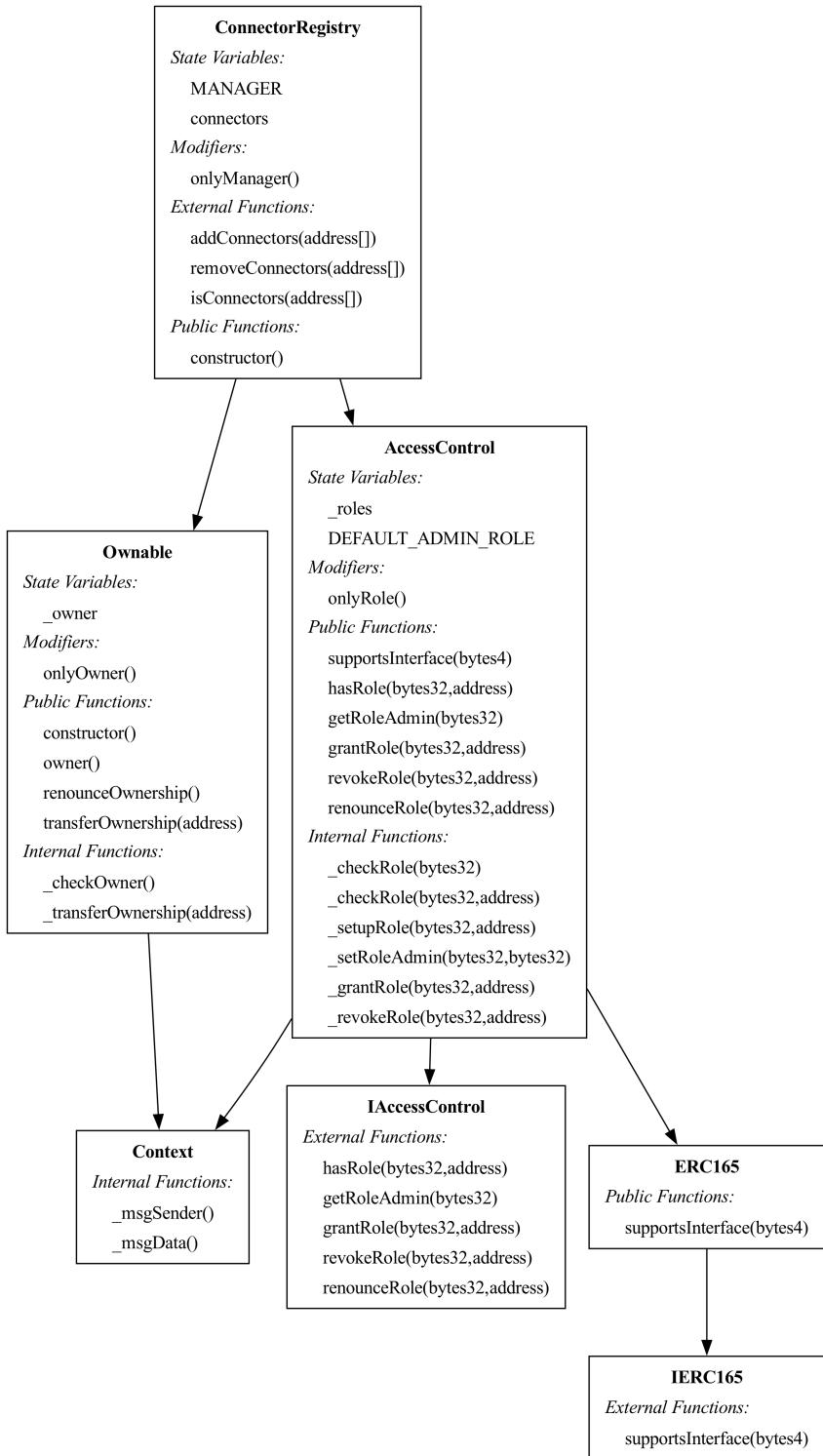
LidoConnector



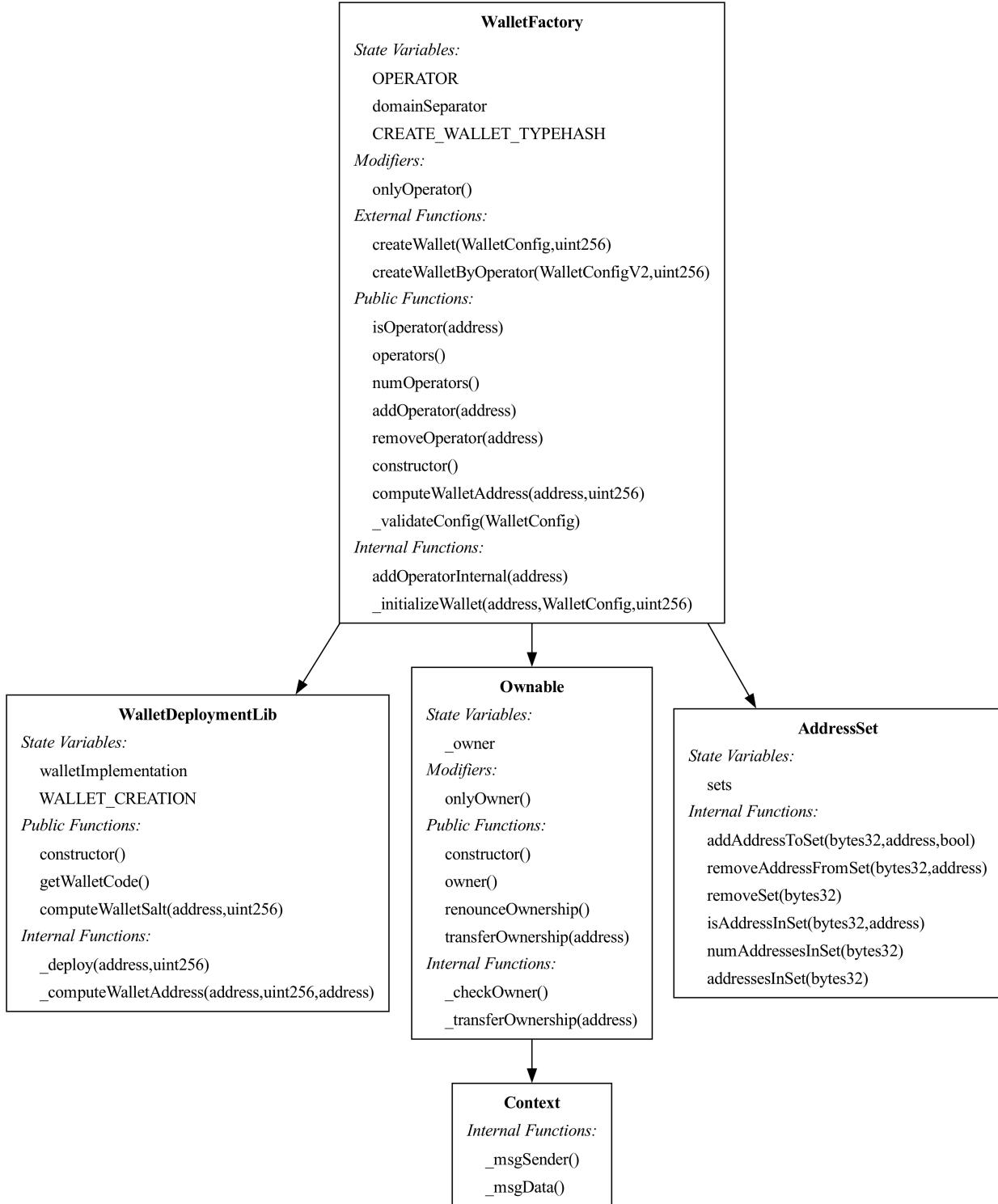
WETHConnector



ConnectorRegistry



WalletFactory



ForwardProxy

ForwardProxy

State Variables:

implManager

Public Functions:

constructor()

Internal Functions:

_implementation()



Proxy

External Functions:

fallback()

receive()

Internal Functions:

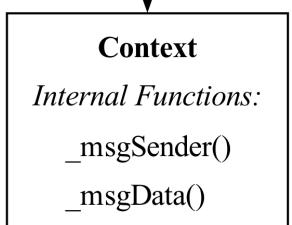
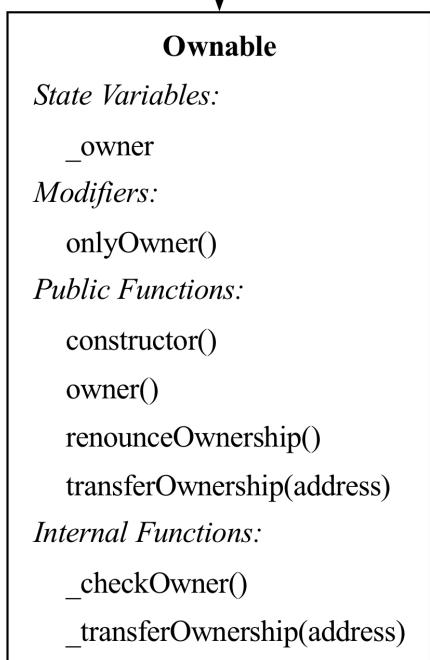
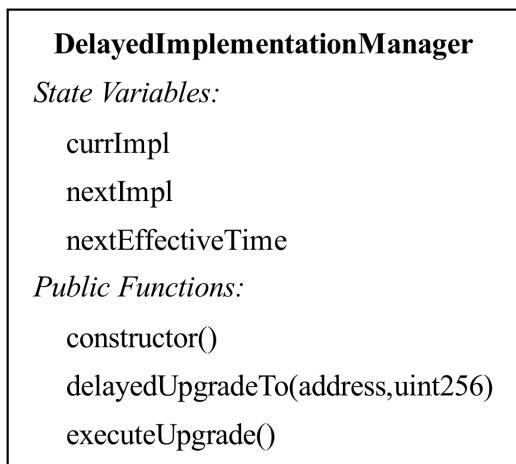
_delegate(address)

_implementation()

_fallback()

_beforeFallback()

DelayedImplementationManager



KyberNetworkPriceOracle

State Variables:

kyber

ETH_ADDR

Public Functions:

constructor()

tokenValue(address,uint256)



PriceOracle

External Functions:

tokenValue(address,uint256)

OwnedMemory

OwnedMemory

State Variables:

master

BROADCAST_ADDR

Modifiers:

isMaster()

Public Functions:

constructor()

getBroadcastAddr(uint256)

setBroadcastAddr(uint256,address)



Memory

State Variables:

mbytes

muint

maddr

Public Functions:

setBytes(uint256,bytes32)

getBytes(uint256)

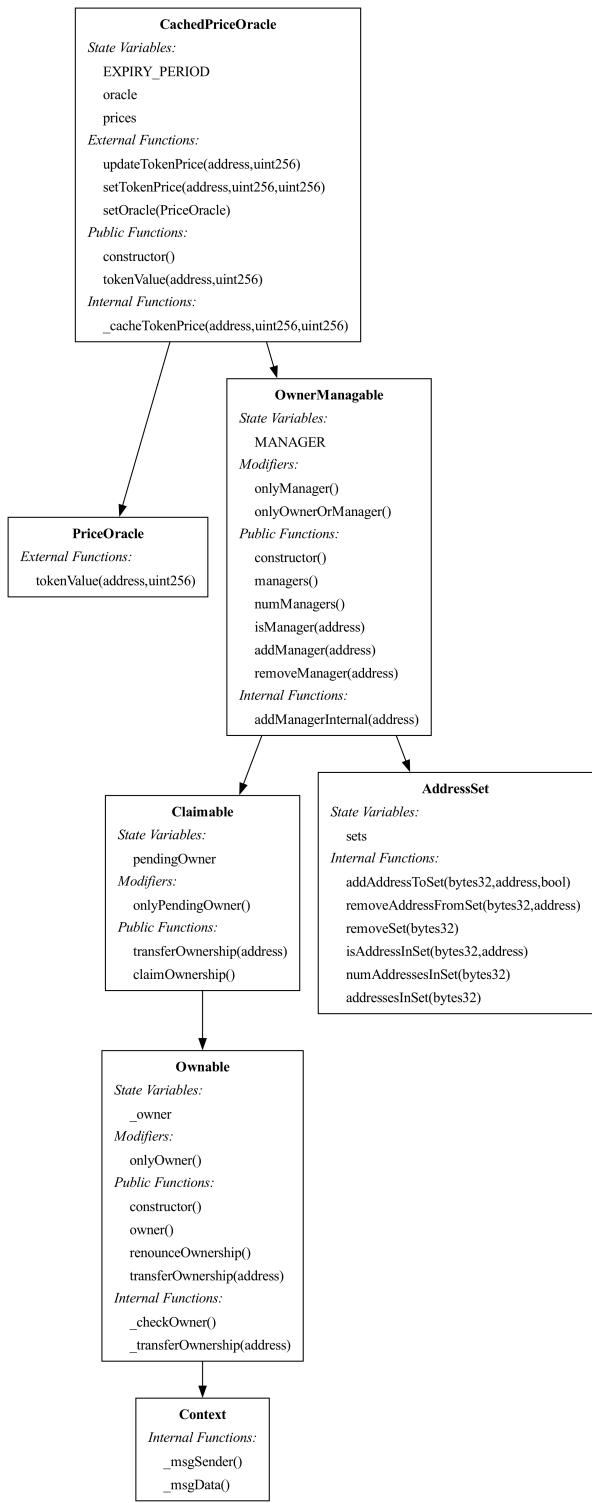
setUint(uint256,uint256)

getUint(uint256)

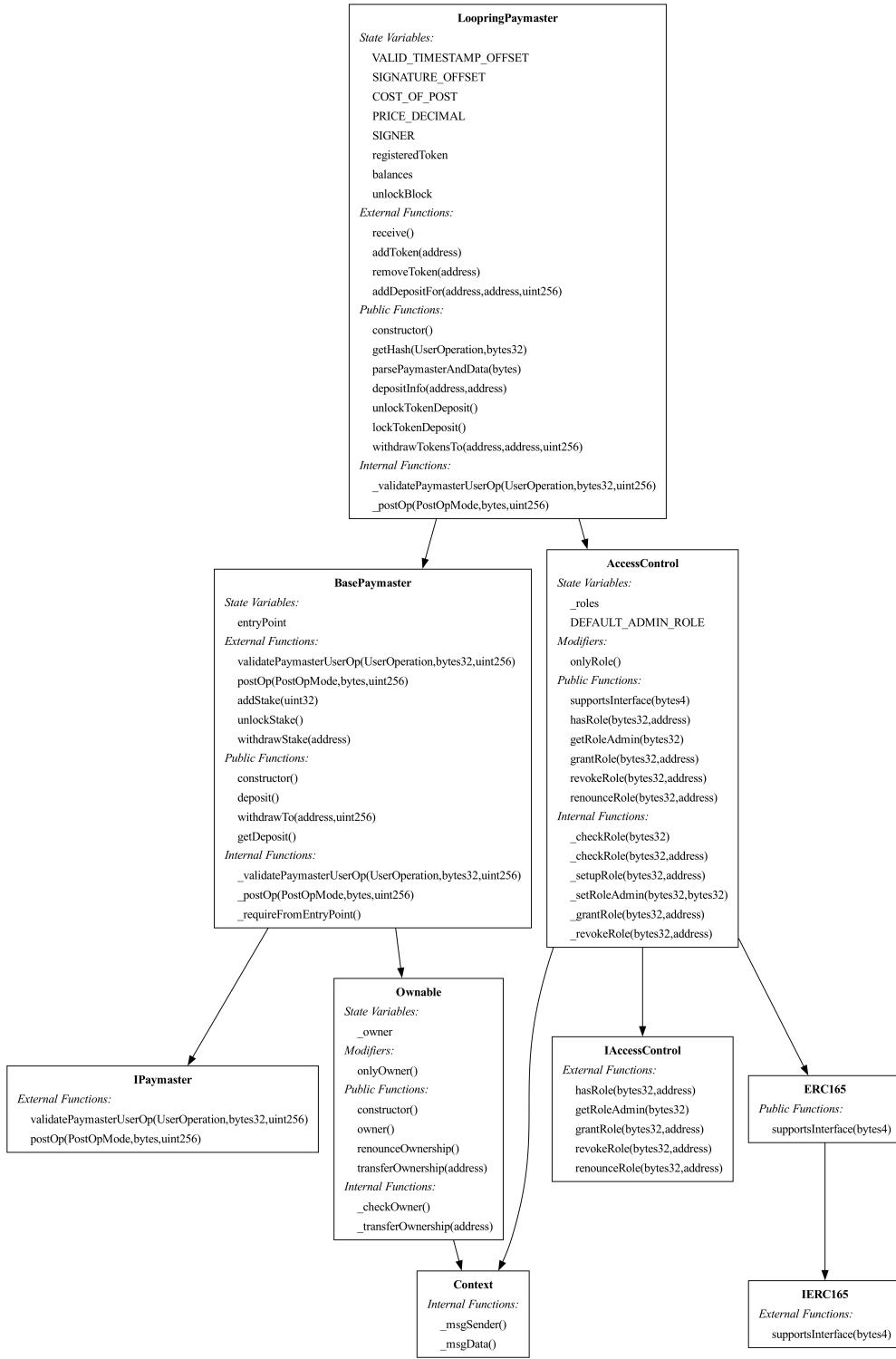
setAddr(uint256,address)

getAddr(uint256)

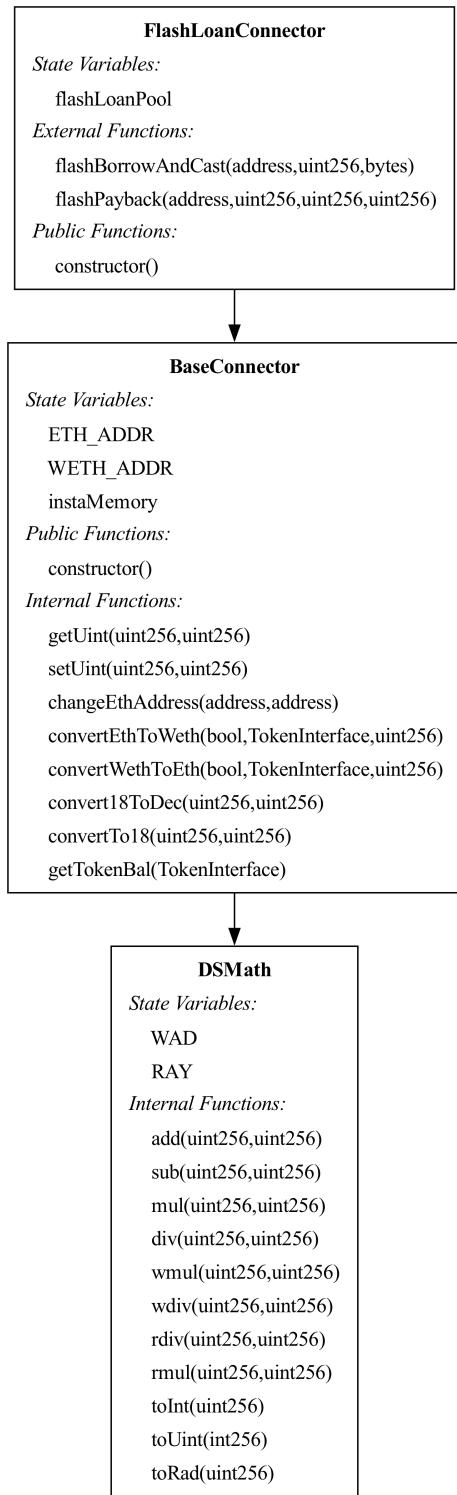
CachedPriceOracle



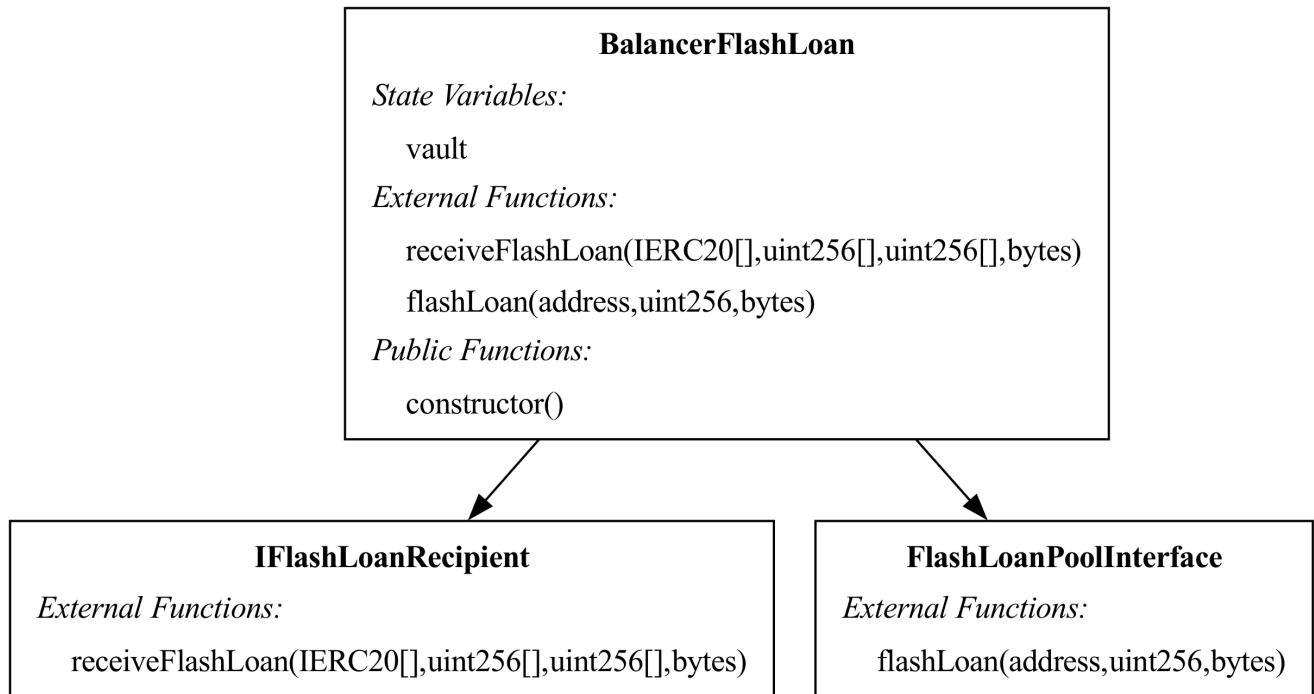
LoopringPaymaster



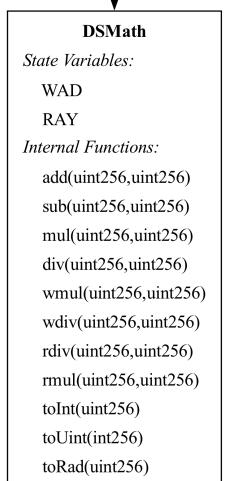
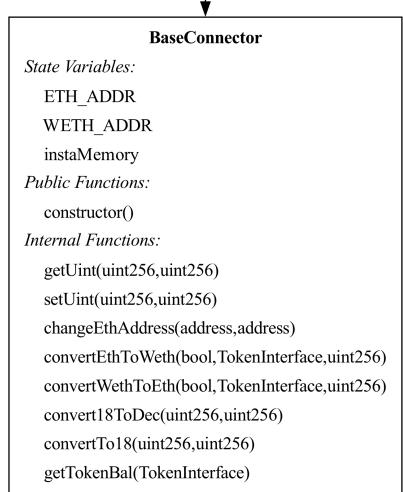
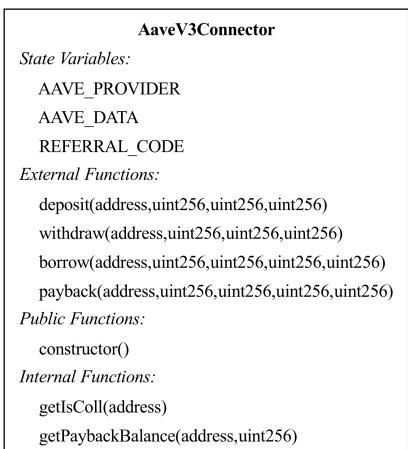
FlashLoanConnector



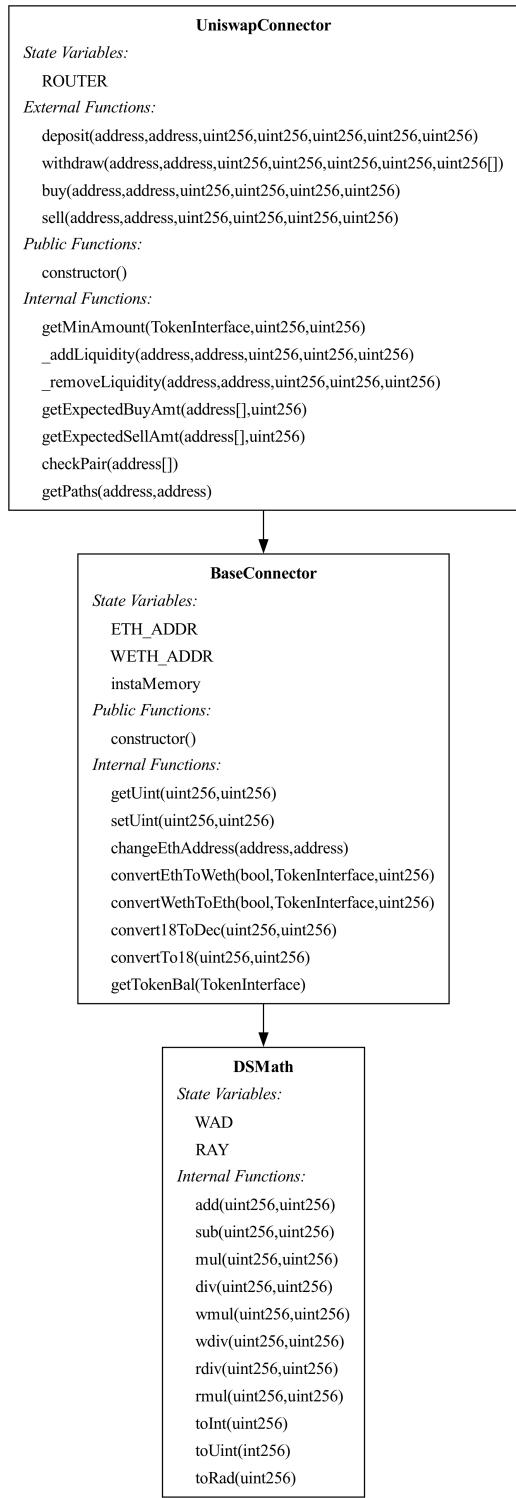
BalancerFlashLoan



AaveV3Connector



UniswapConnector



AggregationalPriceOracle

State Variables:

oracles

Public Functions:

constructor()

tokenValue(address,uint256)



PriceOracle

External Functions:

tokenValue(address,uint256)

CompoundConnector

CompoundConnector

State Variables:

COMP_TROLLER

Public Functions:

constructor()
deposit(address,address,uint256,uint256,uint256)
withdraw(address,address,uint256,uint256,uint256)
borrow(address,address,uint256,uint256,uint256)
payback(address,address,uint256,uint256,uint256)
depositCToken(address,address,uint256,uint256,uint256)
withdrawCToken(address,address,uint256,uint256,uint256)
liquidate(address,address,address,address,address,uint256,uint256,uint256)

Internal Functions:

enterMarket(address)



BaseConnector

State Variables:

ETH_ADDR
WETH_ADDR
instaMemory

Public Functions:

constructor()

Internal Functions:

getUint(uint256,uint256)
setUint(uint256,uint256)
changeEthAddress(address,address)
convertEthToWeth(bool,TokenInterface,uint256)
convertWethToEth(bool,TokenInterface,uint256)
convert18ToDec(uint256,uint256)
convertTo18(uint256,uint256)
getTokenBal(TokenInterface)



DSMath

State Variables:

WAD
RAY

Internal Functions:

add(uint256,uint256)
sub(uint256,uint256)
mul(uint256,uint256)
div(uint256,uint256)
wmul(uint256,uint256)
wdiv(uint256,uint256)
rdiv(uint256,uint256)
rmul(uint256,uint256)
toInt(uint256)
toUint(int256)
toRad(uint256)

UniswapV2PriceOracle

State Variables:

factory

wethAddress

Public Functions:

constructor()

tokenValue(address,uint256)

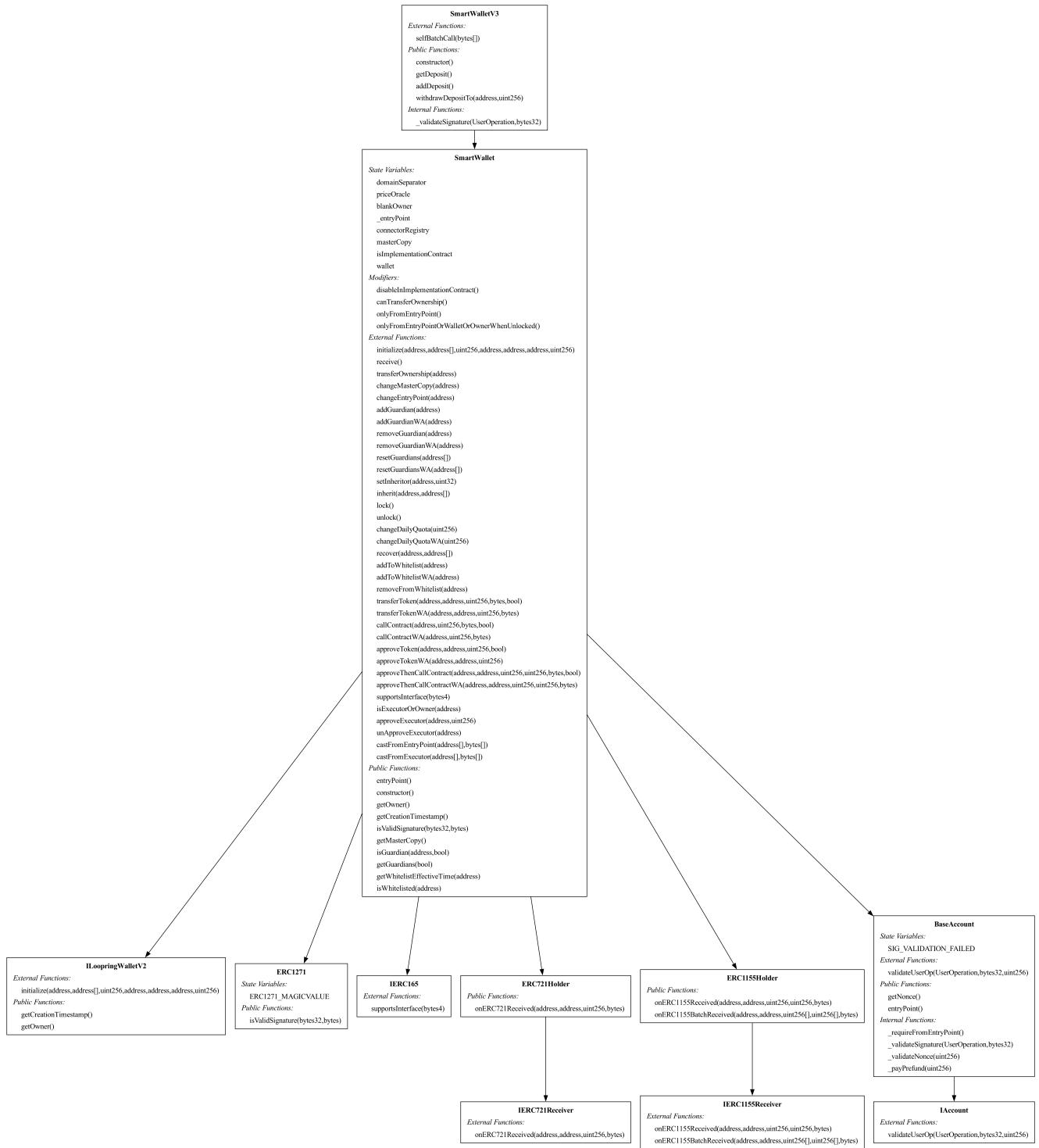


PriceOracle

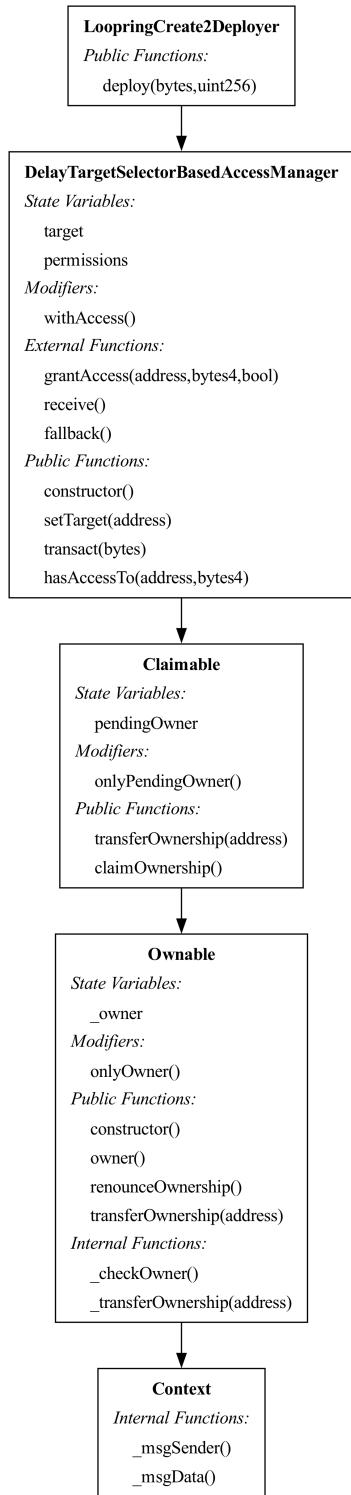
External Functions:

tokenValue(address,uint256)

SmartWalletV3



LoopringCreate2Deployer



OfficialGuardian

