Report

v. 1.0

Customer

Contango

# Smart Contract Audit

# Core

12th March 2023

Report prepared by

ABDK Consulting

# Contents

# 1 Changelog

| # | Date | Author | Description |
|---|---|---|---|
| 0.1 | 11.03.23 | A. Zveryanskaya | Initial Draft |
| 0.2 | 11.03.23 | A. Zveryanskaya | Minor revision |
| 1.0 | 12.03.23 | A. Zveryanskaya | Release |

# 2 Introduction



The following document provides the result of the audit performed by ABDK Consulting (Mikhail Vladimirov and Dmitry Khovratovich) at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.

Contango is a unique decentralized market offering expirables, contracts to buy or sell an asset at a set price and date in the future.

# 3 Project scope

We were asked to review:

- Original Code
- Code with Fixes

Files:

| | | | |
|---|---|---|---|
| **/** | | | |
| | ContangoPositionNFT.sol | ExecutionProcessorLib.sol | PoolORacle.sol |
| **batchable/** | | | |
| | Batchable.sol | PermitForwarder.sol | WethHandler.sol |
| **interfaces/** | | | |
| | IContango.sol | IContangoQuoter.sol | IContangoView.sol |
| | IFeeModel.sol | | |
| **libraries/** | | | |
| | CodecLib.sol | IDataTypes.sol | ErrorLib.sol |
| | MathLib.sol | PositionLib.sol | StorageLib.sol |
| | TransferLib.sol | | |
| **liquiditysource/yield-protocol/** | | | |
| | ContangoYield.sol | Yield.sol | YieldUtils.sol |
| **liquiditysource/** | | | |
| | ContangoBase.sol | SlippageLib.sol | UniswapV3Handler.sol |
| **models/** | | | |
| | FixedFeeModel.sol | | |
| **periphery/** | | | |
| | CashSettler.sol | | |
| **utils/** | | | |
| | Balanceless.sol | | |

# 4 Methodology

The methodology is not a strict formal procedure, but rather a selection of methods and tactics combined differently and tuned for each particular project, depending on the project structure and technologies used, as well as on client expectations from the audit.

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows best code practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places as well as their visibility scopes and access levels are relevant. At this phase, we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and done properly. At this phase, we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check if code actually does what it is supposed to do, if that algorithms are optimal and correct, and if proper data types are used. We also make sure that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

We classify issues by the following severity levels:

- **Critical issue** directly affects the smart contract functionality and may cause a significant loss.
- **Major issue** is either a solid performance problem or a sign of misuse: a slight code modification or environment change may lead to loss of funds or data. Sometimes it is an abuse of unclear code behaviour which should be double checked.
- **Moderate issue** is not an immediate problem, but rather suboptimal performance in edge cases, an obviously bad code practice, or a situation where the code is correct only in certain business flows.
- **Minor issues** contain code style, best practices and other recommendations.

# 5 Our findings

We found 17 major, and a few less important issues. All identified Major issues have been fixed or otherwise addressed in collaboration with the client.

| | Active | Fixed |
|---|---|---|
| Major | 8 | 9 |
| Moderate | 2 | 6 |
| Minor | 22 | 37 |

Fixed 52 out of 84 issues

# 6 Major Issues

## CVF-1. INFO

- **Category** Flaw
- **Source** WethHandler.sol

**Description** This function allows anybody to steal all WETH (and all ether using the "wrapETH" function) from the contract.

**Recommendation** Consider limiting access to this function.

**Client Comment** *This is by design, wrap/unwrap are meant to be used with batched calls and the contango address should never hold any balance. It follow the same pattern of Uniswap's Router (https://arbiscan.io/address/0×68b3465833fb72A70ecDF485E0e4C7bD8665Fc45)*

```
27 function unwrapWETH(address payable to) external payable returns (
      ↪ uint256 unwrapped) {
```

## CVF-2. FIXED

- **Category** Flaw
- **Source** FixedFeeModel.sol

**Description** There is not range check for the argument.

**Recommendation** Consider adding appropriate checks.

**Client Comment** *Fixed and merged.*

```
10 constructor(uint256 _fee) {
```

## CVF-3. INFO

- **Category** Suboptimal
- **Source** ContangoPositionNFT.sol

**Description** There is an efficient way to trim an in-memory array via assembly.

**Recommendation** Consider returning an array without trailing zeros.

**Client Comment** *Already marked for deprecation, as this data will come from the graph, so not fixing it.*

```
66 /// PositionId == 0 is always invalid, so as soon it shows up in the
   ↪  array is safe to assume the rest of it is empty
```

## CVF-4. FIXED

- **Category** Procedural
- **Source** PoolOracle.sol

**Description** Using “require” statements with side effects is a bad practice, that makes code more error-prone and harder to read.

**Recommendation** Consider checks from assignments.

**Client Comment** *Fixed and PR’d on yield repo.*

```
57 require((periodSize = windowSize_ / granularity_) * granularity_ ==
   ↪ windowSize_, "WINDOW_NOT_EVENLY_DIVISIBLE");
```

## CVF-5. FIXED

- **Category** Suboptimal
- **Source** PoolOracle.sol

**Description** The expression “poolObservations[pool]” is calculated multiple times.

**Recommendation** Consider calculating once and reusing.

**Client Comment** *Fixed and PR’d on yield repo.*

```
75 uint256 length = poolObservations[pool].length;
```

```
88         o = poolObservations[pool][oldestObservationIndex];
```

## CVF-6. FIXED

- **Category** Suboptimal
- **Source** PoolOracle.sol

**Description** This looks like waste of gas.

**Recommendation** Consider refactoring to populate observation slots with real observations once they come it.

**Client Comment** *Fixed and PR'd on yield repo.*

```
112 // populate the array with empty observations (only on the first
    ↪ call ever for each pool)
```

## CVF-7. FIXED

- **Category** Suboptimal
- **Source** PoolOracle.sol

**Recommendation** This could be simplified as: require (price >= 0); require (amount » 192 == 0); result = (amount « 64) / uint128 (price);

**Client Comment** *Fixed and PR'd on yield repo.*

```
254 result = amount.divu(WAD).div(price).mulu(WAD); // result = amount /
    ↪  price
```

```
284 result = amount.divu(WAD).div(price).mulu(WAD); // result = amount /
    ↪  price
```

## CVF-8. INFO

- **Category** Unclear behavior
- **Source** CashSettler.sol

**Description** This function doesn't allow transferring plain ether.

**Recommendation** Consider adding such ability.

**Client Comment** *Not necessary since only WETH is used on instruments and ETH can only be sent to this contract by unwrapping WETH.*

```
123 function _transferEquity(ERC20 token, address to) internal returns (
    ↪ uint256 balance) {
```

## CVF-9. INFO

- **Category** Unclear behavior
- **Source** Batchable.sol

**Description** Reverting with a named error here makes it impossible to distinguish the following two situations: 1. An inner transaction reverted with no data 2. An inner transaction reverted with the “TransactionRevertedSilently” error.

**Recommendation** Consider reverting with empty data in case an inner transaction reverted with empty data.

**Client Comment** *This error is only used here so it can only mean an inner revert with no data.*

```
25 if (returnData.length == 0) revert TransactionRevertedSilently();
```

## CVF-10. INFO

- **Category** Unclear behavior
- **Source** Batchable.sol

**Description** An error data returned here doesn’t contain the index of the failed transaction within the batch.

**Recommendation** Consider reverting with a named error encapsulating the index of a failed. transaction plus the data returned from the failed transaction. Also consider including the data returned by all the preceding successful transactions as such data could simplify debugging.

**Client Comment** *Not trivial to revert the underlying data + index via assembly and keep it readable. With current tools, the dev can either get stack traces via foundry or debug the tx on tenderly, so not fixing it.*

```
28 revert(add(32, returnData), mload(returnData))
```

## CVF-11. INFO

- **Category** Documentation
- **Source** Yield.sol

**Description** The business logic of this function is quite complicated.

**Recommendation** Consider describing it in a documentation comment.

**Client Comment** *The functions are well documented already, unclear what can be improved.*

```
192 function completeOpen(UniswapV3Handler.Callback memory callback)
      ↪ internal {
```

## CVF-12. INFO

- **Category** Procedural
- **Source** YieldUtils.sol

**Description** This looks like a piece of business logic separated from the function where it should be and hidden in an unexpected place.

**Recommendation** Consider moving this logic into the function that calls the “maxFYTokenOut” function.

**Client Comment** *This encapsulates issues on a 3rd party contract and we want to keep encapsulated here.*

```
35 } else if (f.selector == IPool.maxFYTokenOut.selector) {
```

## CVF-13. FIXED

- **Category** Bad naming
- **Source** PositionLib.sol

**Description** The function signature looks like the function just returned the owner for a positions, while actually the function verifies that a position is owner by the message sender, and reverts otherwise.

**Recommendation** Consider clearly documenting this behavior and also renaming the function.

**Client Comment** *Fixed and merged.*

```
12 function positionOwner(PositionId positionId) internal view returns
      ↪ (address trader) {
```

## CVF-14. FIXED

- **Category** Procedural
- **Source** IContangoQuoter.sol

**Description** Specifying a particular compiler version makes it harder to upgrade to newer versions. Also it makes is harder to use this interface in other projects.

**Recommendation** Consider specifying as "^0.8.0".

**Client Comment** *Fixed and merged.*

```
2 pragma solidity 0.8.17;
```

## CVF-15. INFO

- **Category** Bad datatype
- **Source** IContangoQuoter.sol

**Description** These functions look like they don't modify the blockchain state.

**Recommendation** Consider declaring them as "view".

**Client Comment** *Limited by the uniswap quoter not being view, should be fine since it's inteded for off-chain use and these can do callstatic.*

```
12 function positionStatus(PositionId positionId, uint24 uniswapFee)
      ↪ external returns (PositionStatus memory);

18 function openingCostForPosition(OpeningCostParams calldata params)
      ↪ external returns (ModifyCostResult memory);

23 function modifyCostForPosition(ModifyCostParams calldata params)
      ↪ external returns (ModifyCostResult memory);

28 function deliveryCostForPosition(PositionId positionId) external
      ↪ returns (uint256);
```

## CVF-16. FIXED

- **Category** Procedural
- **Source** IContangoView.sol

**Description** Specifying a particular compiler version makes it harder to upgrade to newer versions. Also it makes is harder to use this interface in other projects.

**Recommendation** Consider specifying as "^0.8.0".

**Client Comment** *Fixed and merged.*

```
2 pragma solidity 0.8.17;
```

## CVF-17. FIXED

- **Category** Procedural
- **Source** IContango.sol

**Description** Specifying a particular compiler version makes it harder to upgrade to newer versions. Also it makes is harder to use this interface in other projects.

**Recommendation** Consider specifying as "^0.8.0".

**Client Comment** *Fixed and merged.*

```
2 pragma solidity 0.8.17;
```

# 7 Moderate Issues

## CVF-18. INFO

- **Category** Unclear behavior
- **Source** ContangoPositionNFT.sol

**Description** This allows a minter to burn other people's NFTs with out explicit approval, which is dangerous.

**Recommendation** Consider limiting this ability, e.g. requiring an NFT to be explicitly approved to the miner, or even owned by the miner.

**Client Comment** *The app needs to be able to mint and burn position NFTs, so not sure what else we can do apart from allowing it via a role, so we're safe to upgrade to a different address if need be (although not expected because it's already under a proxy).*

```
35 function burn(PositionId positionId) external onlyRole(MINTER) {
```

## CVF-19. FIXED

- **Category** Unclear behavior
- **Source** PoolOracle.sol

**Recommendation** It should be "length" instead of "granularity". Otherwise array index out of bounds error could happen in the next statement.

**Client Comment** *Fixed and PR'd on yield repo.*

```
85 uint256 oldestObservationIndex = (++observationIndex) % granularity;
```

## CVF-20. FIXED

- **Category** Overflow/Underflow
- **Source** PoolOracle.sol

**Description** Multiplying by “WAD” and then dividing by “RAY” significantly increases probability of phantom overflow.

**Recommendation** Consider calculating as: (currentCumulativeRatio_ - oldestObservation.ratioCumulative) / timeElapsed / (RAY / WAD)

**Client Comment** *Fixed and PR’d on yield repo.*

```
158 twar = ((currentCumulativeRatio_ - oldestObservation.ratioCumulative
    ↪ ) * WAD) / (timeElapsed * RAY);
```

## CVF-21. FIXED

- **Category** Overflow/Underflow
- **Source** ExecutionProcessorLib.sol

**Description** Overflow is possible here.

**Recommendation** Consider using safe conversion.

**Client Comment** *Fixed and merged.*

```
99  openCost = uint256(int256(openCost) + cost);
```

```
124     positionCollateral = collateralDelta - int256(fee);
```

```
161     pnl = int256(cost) - int256(closedCost);
```

```
218 int256 pnl = int256(cost) - int256(closedCost);
```

```
242     collateral = collateral - int256(fee);
```

## CVF-22. FIXED

- **Category** Unclear behavior
- **Source** UniswapV3Handler.sol

**Description** In case the pool doesn't exist, this will call a non-contract address and silently do nothing.

**Recommendation** Consider reverting in such a case.

**Client Comment** *Fixed and merged.*

```
54 IUniswapV3Pool(UNISWAP_FACTORY.computeAddress(poolKey)).swap({
```

## CVF-23. FIXED

- **Category** Overflow/Underflow
- **Source** UniswapV3Handler.sol

**Description** Overflow is possible here.

**Recommendation** Consider using safe conversion.

**Client Comment** *Fixed and merged.*

```
57 amountSpecified: baseForQuote ? int256(callback.fill.hedgeSize) : -
   ↪ int256(callback.fill.hedgeSize),
```

## CVF-24. FIXED

- **Category** Overflow/Underflow
- **Source** Yield.sol

**Description** Overflow is possible here.

**Recommendation** Consider using safe conversion.

**Client Comment** *Fixed and merged.*

```
58  _open(symbol, positionId, trader, instrument, quantity, limitCost,
        ↪ int256(collateral), payer, lendingLiquidity);

183     int256(collateral),

212     callback.fill.collateral = SignedMath.min(callback.fill.
            ↪ collateral, int256(callback.fill.hedgeCost));

218 uint128 amountToBorrow = (int256(callback.fill.hedgeCost) - callback
        ↪ .fill.collateral).toUint256().toUint128();

233     int128(ink), // Use the fyTokens we bought using the flash swap
            ↪ as ink (collateral)
        int128(art) // Amount to borrow in future value

323         art = -int128(balances.art);

366             art = -int128(

552     -int256(art).toInt128() // We burn all the (fy)Quote we just
            ↪ bought

556 int256 cost = -int256(art - collateral);

584 int256 cost = int256(art - collateral);

648     uint128 maxBaseIn = uint128(lendingLiquidity);
```

## CVF-25. INFO

- **Category** Flaw
- **Source** PositionLib.sol

**Description** This check looks very dangerous. It doesn't allow arbitrary payers, but allows to use one valid payer instead of another.

**Recommendation** Consider replacing with more specific checks relevant to particular use cases.

**Client Comment** *This is by design, we don't want people to get arbitrary wallets to pay for their position.*

```
73 if (payer != trader && payer != address(this) && payer != msg.sender
   ↪ ) {
```

# 8 Minor Issues

## CVF-26. INFO

- **Category** Procedural
- **Source** FixedFeeModel.sol

**Description** Specifying a particular compiler version makes it harder to upgrade to newer versions.

**Recommendation** Consider specifying as "^0.8.0". Also relevant for: ContangoPositionNFT.sol, Balanceless.sol, ContangoBase.sol, ExecutionProcessorLib.sol, SlippageLib.sol, UniswapV3Handler.sol, ErrorLib.sol, Batchable.sol, Yield.sol, YieldUtils.sol, ContangoYield.sol, TransferLib.sol, PositionLib.sol, MathLib.sol, WethHandler.sol, PermitForwarder.sol, StorageLib.sol, DataTypes.sol, IFeeModel.sol.

**Client Comment** *Not fixing it until we can ensure the very same compiler version is used on dev/test and deployment.*

```
2 pragma solidity 0.8.17;
```

## CVF-27. FIXED

- **Category** Procedural
- **Source** PoolOracle.sol

**Description** This version requirement is inconsistent with other files. Also, it allows future major release (0.9+), while compatibility with such release cannot be guaranteed.

**Recommendation** Consider specifying as "^0.8.0".

**Client Comment** *Fixed and PR'd on yield repo.*

```
2 pragma solidity >=0.8.15;
```

## CVF-28. FIXED

- **Category** Procedural
- **Source** PoolOracle.sol

**Description** We didn't review these files.

```
4 import "../interfaces/IPoolOracle.sol";
  import {Exp64x64} from "../Exp64x64.sol";
  import {Math64x64} from "../Math64x64.sol";
```

## CVF-29. INFO

- **Category** Unclear behavior
- **Source** PoolOracle.sol

**Description** This contract is not used in other files. Its role is unclear.

**Client Comment** *Nothing to fix here.*

```
15 contract PoolOracle is IPoolOracle {
```

## CVF-30. INFO

- **Category** Bad naming
- **Source** PoolOracle.sol

**Recommendation** Events are usually named via nouns, such as “Observation”.

**Client Comment** *This repo has mixed use on event naming and we believe events should be actions that happened in the past, hence this naming. not fixing it.*

```
19 event ObservationRecorded(IPool indexed pool, uint256 index,
      ↪ Observation observation);
```

## CVF-31. INFO

- **Category** Suboptimal
- **Source** PoolOracle.sol

**Recommendation** Consider passing “periodSize” as an argument instead of “windowSize”, and calculating “windowSize” in the constructor. Such approach would make this check unnecessary.

**Client Comment** *We actually want to keep the check in case the deployment uses bad values.*

```
57 require((periodSize = windowSize_ / granularity_) * granularity_ ==
      ↪ windowSize_, "WINDOW_NOT_EVENLY_DIVISIBLE");
```

## CVF-32. FIXED

- **Category** Suboptimal
- **Source** PoolOracle.sol

**Description** The expression “poolObservations[pool]” is calculated twice.

**Recommendation** Consider calculating once and reusing.

**Client Comment** *Fixed and PR’d on yield repo.*

```
114     for (uint256 i = poolObservations[pool].length; i < granularity;
          ↪  ) {
```

```
122 Observation storage observation = poolObservations[pool][index];
```

## CVF-33. FIXED

- **Category** Suboptimal
- **Source** PoolOracle.sol

**Recommendation** The conversion to “IPool” is redundant as “pool” is already “IPool”.

**Client Comment** *Fixed and PR’d on yield repo.*

```
127 (observation.ratioCumulative, observation.timestamp) = IPool(pool).
      ↪ currentCumulativeRatio();
```

## CVF-34. INFO

- **Category** Suboptimal
- **Source** PoolOracle.sol

**Description** The returned values are ignored here.

**Recommendation** Consider returning a bit mask of “updated” flags for individual pools.

**Client Comment** *This was a concious design choice. not fixing it.*

```
137 updatePool(pools[i]);
```

## CVF-35. INFO

- **Category** Bad naming
- **Source** Balanceless.sol

**Recommendation** Events are usually named via nouns, such as “BalanceCollection”.

**Client Comment** *Events should be named as actions that happened in the past, so we’ll keep this naming convention and it is consistently used throughout the codebase, not fixing it.*

```
10 event BalanceCollected(address indexed token, address indexed to,
      ↪ uint256 amount);
```

## CVF-36. FIXED

- **Category** Bad datatype
- **Source** Balanceless.sol

**Recommendation** The type of the “token” parameter should be “IERC20”.

**Client Comment** *Fixed and merged.*

```
10 event BalanceCollected(address indexed token, address indexed to,
      ↪ uint256 amount);
```

## CVF-37. FIXED

- **Category** Bad datatype
- **Source** Balanceless.sol

**Recommendation** The type of the “token” argument should be “IERC20”.

**Client Comment** *Fixed and merged.*

```
13 function _collectBalance(address token, address payable to, uint256
      ↪ amount) internal {
```

## CVF-38. FIXED

- **Category** Procedural
- **Source** CashSettler.sol

**Description** We didn't review this file.

```
8 import "../dependencies/Balancer.sol";
```

## CVF-39. INFO

- **Category** Bad naming
- **Source** CashSettler.sol

**Recommendation** Events are usually named via nouns, such as "Settlement".

**Client Comment** *Events should be named as actions that happened in the past, so we'll keep this naming convention and it is consistently used throughout the codebase, not fixing it.*

```
19 event PositionSettled(
```

## CVF-40. INFO

- **Category** Suboptimal
- **Source** CashSettler.sol

**Description** Hardcoding mainnet addresses is a bad practice, as it makes testing harder.

**Recommendation** Consider passing the balancer address as a constructor argument and storing in an immutable variable.

**Client Comment** *Balancer has the same address for the vault on all networks, so it has the same effect of immutable variables with the upside of being crystal clear which address it is pointing too without needing to query the deployed contract.*

```
50 IFlashLoaner public constant BALANCER = IFlashLoaner(0
   ↪ xBA12222222228d8Ba445958a75a0704d566BF2C8);
```

## CVF-41. FIXED

- **Category** Bad datatype
- **Source** ContangoBase.sol

**Recommendation** The type of the “token” argument should be “IERC20”.

**Client Comment** *Fixed and merged.*

```
75 function setTrustedToken(address token, bool trusted) external
   ↪ onlyRole(DEFAULT_ADMIN_ROLE) {
```

```
83 function collectBalance(address token, address payable to, uint256
   ↪ amount) external onlyRole(DEFAULT_ADMIN_ROLE) {
```

## CVF-42. FIXED

- **Category** Suboptimal
- **Source** ContangoBase.sol

**Description** This function is redundant, as a contract without fallback function would revert on unknown function selectors.

**Recommendation** Consider removing this function.

**Client Comment** *Fixed and merged.*

```
116 fallback() external payable {
```

## CVF-43. INFO

- **Category** Bad naming
- **Source** ExecutionProcessorLib.sol

**Recommendation** Events are usually named via nouns, such as “Position”, “PositionLiquidation”, “PositionClose” etc.

**Client Comment** *Events should be named as actions that happened in the past, so we’ll keep this naming convention and it is consistently used throughout the codebase, not fixing it.*

```
20 event PositionUpserted(
```

```
32 event PositionLiquidated(
```

```
42 event PositionClosed(
```

```
54 event PositionDelivered(
```

## CVF-44. INFO

- **Category** Suboptimal
- **Source** ExecutionProcessorLib.sol

**Description** These arguments are don’t affect function logic and are only used when emitting event.

**Recommendation** Consider removing these arguments, and emitting the event from the calling code.

**Client Comment** *The events are under the libs because these are meant to be shared with future underlying protocol integrations, e.g. Notional, and pulling the events would break this encapsulation.*

```
70 Symbol symbol,
```

```
72 address trader,
   uint256 deliverableQuantity,
   uint256 deliveryCost,
```

```
77 address to
```

## CVF-45. FIXED

- **Category** Procedural
- **Source** UniswapV3Handler.sol

**Description** We didn't review this file.

```
7 import "../dependencies/Uniswap.sol";
```

## CVF-46. INFO

- **Category** Suboptimal
- **Source** UniswapV3Handler.sol

**Description** Hardcoding mainnet addresses is a bad practice, as it makes it harder to test code.

**Recommendation** Consider passing Uniswap factory address as an argument to the functions that need it. In calling contracts, consider passing it as a constructor argument and storing in an immutable variable.

**Client Comment** *Uniswap has the same address on all networks, so it has the same effect of immutable variables with the upside of being crystal clear which address it is pointing too without needing to query the deployed contract. Also this is a library, so it'd have to be an extra argument.*

```
35 address internal constant UNISWAP_FACTORY = 0
   ↪ x1F98431c8aD98523631AE4a59f267346ea31F984;
```

## CVF-47. INFO

- **Category** Bad datatype
- **Source** UniswapV3Handler.sol

**Recommendation** The type of this variable should be "IUniswapV3Factory".

**Client Comment** *The library copied from uniswap expects an address, so we avoid the extra casting.*

```
35 address internal constant UNISWAP_FACTORY = 0
   ↪ x1F98431c8aD98523631AE4a59f267346ea31F984;
```

## CVF-48. FIXED

- **Category** Bad naming
- **Source** ErrorLib.sol

**Description** Despite the name this file doesn't contain a library.

**Recommendation** Consider renaming the file to "Errors.sol".

**Client Comment** *Fixed and merged.*

```
2 pragma solidity 0.8.17;
```

## CVF-49. INFO

- **Category** Suboptimal
- **Source** ErrorLib.sol

**Recommendation** The 'positionId' fields should probably be indexed.

**Client Comment** *Can only index event fields.*

```
18 error InvalidPayer(PositionId positionId, address payer);

20 error InvalidPosition(PositionId positionId);

22 error InvalidPositionDecrease(PositionId positionId, int256
   ↪ decreaseQuantity, uint256 currentQuantity);

26 error NotPositionOwner(PositionId positionId, address msgSender,
   ↪ address actualOwner);

28 error PositionActive(PositionId positionId, uint32 maturity, uint256
   ↪  timestamp);

30 error PositionExpired(PositionId positionId, uint32 maturity,
   ↪ uint256 timestamp);
```

## CVF-50. FIXED

- **Category** Procedural
- **Source** Yield.sol

**Description** We didn't review these files.

```
4 import {IPool} from "@yield-protocol/yieldspace-tv/src/interfaces/
     ↪ IPool.sol";
  import {ILadle} from "@yield-protocol/vault-v2/contracts/interfaces/
     ↪ ILadle.sol";
  import {ICauldron} from "@yield-protocol/vault-v2/contracts/
     ↪ interfaces/ICauldron.sol";
  import {IFYToken} from "@yield-protocol/vault-v2/contracts/
     ↪ interfaces/IFYToken.sol";
  import {DataTypes} from "@yield-protocol/vault-v2/contracts/
     ↪ interfaces/DataTypes.sol";
  import {IContangoLadle} from "@yield-protocol/vault-v2/contracts/
     ↪ other/contango/interfaces/IContangoLadle.sol";
```

## CVF-51. INFO

- **Category** Bad naming
- **Source** Yield.sol

**Recommendation** Events are usually named via nouns, such as "Trade", "AddedCollateral", or "RemovedCollateral".

**Client Comment** *Events should be named as actions that happened in the past, so we'll keep this naming convention and it is consistently used throughout the codebase, not fixing it.*

```
28 event ContractTraded(Symbol indexed symbol, address indexed trader,
     ↪ PositionId indexed positionId, Fill fill);
   event CollateralAdded(
```

```
32 event CollateralRemoved(
```

## CVF-52. FIXED

- **Category** Suboptimal
- **Source** Yield.sol

**Description** This syntax looks odd.

**Recommendation** Consider rewriting as: PositionLib.loadActivePosition (positionId, uniswapFee).

**Client Comment** *Fixed and merged.*

```
75 positionId.loadActivePosition(uniswapFee);
```

## CVF-53. FIXED

- **Category** Suboptimal
- **Source** Yield.sol

**Recommendation** This code should be moved into the “else” branch above. This would make the first part of the condition unnecessary.

**Client Comment** *Fixed and merged.*

```
110 if (quantity < 0 && uint256(-quantity) == openQuantity) {
        _deletePosition(positionId);
    }
```

## CVF-54. FIXED

- **Category** Suboptimal
- **Source** Yield.sol

**Description** Such assumptions make code much more error prone, as changes in other parts of the code could easily break them.

**Recommendation** Consider using safe conversion anyway.

**Client Comment** *Fixed and merged.*

```
195 // Cast is safe as the number was previously casted as uint128
```

## CVF-55. FIXED

- **Category** Bad naming
- **Source** Yield.sol

**Description** The function name looks like a name of a getter, while actually this function does modify the state.

**Recommendation** Consider renaming.

**Client Comment** *Fixed and merged.*

```
640 function _getFYTokensToBurn(
```

## CVF-56. FIXED

- **Category** Procedural
- **Source** YieldUtils.sol

**Description** We didn't review these files.

```
4 import {IPool} from "@yield-protocol/yieldspace-tv/src/interfaces/
    ↪ IPool.sol";
```

## CVF-57. FIXED

- **Category** Unclear behavior
- **Source** YieldUtils.sol

**Description** This assignment should be done only if liquidity is not zero.

**Client Comment** *Fixed and merged.*

```
28 IPool pool = IPool(f.address);
```

## CVF-58. FIXED

- **Category** Suboptimal
- **Source** YieldUtils.sol

**Recommendation** The values used here should be named constants.

**Client Comment** *Fixed and merged.*

```
33 if (scaleFactor == 1 && liquidity <= 1e13 || scaleFactor == 1e12 &&
   ↪ liquidity <= 1e3) {
```

## CVF-59. FIXED

- **Category** Procedural
- **Source** ContangoYield.sol

**Description** We didn't review these files.

```
6 import {IContangoLadle} from "@yield-protocol/vault-v2/contracts/
   ↪ other/contango/interfaces/IContangoLadle.sol";
import "@yield-protocol/vault-v2/contracts/other/contango/interfaces
   ↪ /IContangoWitchListener.sol";
```

## CVF-60. FIXED

- **Category** Unclear behavior
- **Source** TransferLib.sol

**Description** Including both addresses into an error, while only one of the addresses is problematic look odd.

**Recommendation** Consider declaring two separate errors: "ZeroPayer()" and "ZeroDestination()".

**Client Comment** *Fixed and merged.*

```
13 revert ZeroAddress(payer, to);
```

## CVF-61. INFO

- **Category** Suboptimal
- **Source** PositionLib.sol

**Recommendation** The returned value is redundant, as it is always equals to “msg.sender”.

**Client Comment** *Convenient for the caller, not fixing it.*

```
12 function positionOwner(PositionId positionId) internal view returns
   ↪ (address trader) {
```

## CVF-62. FIXED

- **Category** Documentation
- **Source** PositionLib.sol

**Description** This assignment looks like a hack.

**Recommendation** Consider refactoring to avoid such things or at least clearly document them.

**Client Comment** *Fixed and merged.*

```
60 instrument.uniswapFeeTransient = uniswapFee;
```

## CVF-63. INFO

- **Category** Suboptimal
- **Source** PositionLib.sol

**Recommendation** The “owner” returned value is redundant, as it always equals to “msg.sender”.

**Client Comment** *Convenient for the caller, not fixing it.*

```
66 returns (uint256 openQuantity, address owner, Symbol symbol,
   ↪ Instrument memory instrument)
```

## CVF-64. INFO

- **Category** Suboptimal
- **Source** MathLib.sol

**Recommendation** This branch should be the first one for efficiency: if (fromPrecision == toPrecision) { scale = value; } else if (fromPrecision > toPrecision) { ... } else { ... }

**Client Comment** *This was included by mistake, it's inteded to use with Notional integration where they have an internal precision of 8 decimals and most tokens are not 8 decimals, not fixing it.*

```
17 } else {
       scaled = value;
   }
```

## CVF-65. FIXED

- **Category** Suboptimal
- **Source** CodecLib.sol

**Recommendation** This conditions could be simplified as: if (int128 (n) != n)

**Client Comment** *Fixed and merged.*

```
9 if (n > type(int128).max || n < type(int128).min) {
```

## CVF-66. FIXED

- **Category** Suboptimal
- **Source** CodecLib.sol

**Recommendation** This condition could be simplified as: if (uint128 (n) != n)

**Client Comment** *Fixed and merged.*

```
16 if (n > type(uint128).max) {
```

## CVF-67. FIXED

- **Category** Suboptimal
- **Source** CodecLib.sol

**Recommendation** This could be simplified as: encoded = a « 128 | b;

**Client Comment** *Fixed and merged.*

```
23 encoded |= uint256(uint128(a)) << 128;
   encoded |= uint256(uint128(b));
```

## CVF-68. FIXED

- **Category** Suboptimal
- **Source** CodecLib.sol

**Recommendation** This could be simplified as: encoded = uint256(a) « 128 | uint128(int128(b))

**Client Comment** *Fixed and merged.*

```
33 encoded |= uint256(uint128(int128(a))) << 128;
   encoded |= uint256(uint128(int128(b)));
```

## CVF-69. FIXED

- **Category** Bad datatype
- **Source** WethHandler.sol

**Recommendation** The type of the “WETH” parameter should be “IWETH9”.

**Client Comment** *Fixed and merged.*

```
10 error OnlyFromWETH(address weth, address sender);
```

## CVF-70. INFO

- **Category** Suboptimal
- **Source** WethHandler.sol

**Recommendation** The “weth” parameter is redundant, as its value could be obtained from a public immutable variable.

**Client Comment** *Useful for debugging.*

```
10 error OnlyFromWETH(address weth, address sender);
```

## CVF-71. FIXED

- **Category** Procedural
- **Source** PermitForwarder.sol

**Description** We didn’t review this file.

```
6 import "dss-interfaces/src/dss/DaiAbstract.sol";
```

## CVF-72. INFO

- **Category** Bad datatype
- **Source** PermitForwarder.sol

**Recommendation** The parameter type should be “IERC20”.

**Client Comment** *This would cause too many castings and make the code less readable on the PermitForwarder, not fixing it.*

```
12 error UnknownToken(address token);
```

## CVF-73. FIXED

- **Category** Procedural
- **Source** StorageLib.sol

**Description** We didn't review these files.

```
7  import {DataTypes} from "@yield-protocol/vault-v2/contracts/
       ↪ interfaces/DataTypes.sol";
   import {IContangoLadle} from "@yield-protocol/vault-v2/contracts/
       ↪ other/contango/interfaces/IContangoLadle.sol";
   import {ICauldron} from "@yield-protocol/vault-v2/contracts/
       ↪ interfaces/ICauldron.sol";
```

## CVF-74. INFO

- **Category** Bad naming
- **Source** StorageLib.sol

**Recommendation** Events are usually named via nouns, such as "FeeModel" or "ClosingOnlyStatus".

**Client Comment** *Events should be named as actions that happened in the past, so we'll keep this naming convention and it is consistently used throughout the codebase, not fixing it.*

```
19 event FeeModelUpdated(Symbol indexed symbol, IFeeModel feeModel);
20 event ClosingOnlySet(Symbol symbol, bool closingOnly);
```

```
138 event YieldInstrumentCreated(Instrument instrument, YieldInstrument
        ↪ yieldInstrument);
    event LadleSet(IContangoLadle ladle);
140 event CauldronSet(ICauldron cauldron);
```

```
263 event TreasurySet(address treasury);
    event PositionNFTSet(address positionNFT);
    event ClosingOnlySet(bool closingOnly);
    event TokenTrusted(address indexed token, bool trusted);
    event ProxyHashSet(bytes32 proxyHash);
```

## CVF-75. FIXED

- **Category** Suboptimal
- **Source** StorageLib.sol

**Recommendation** The “symbol” parameter should be indexed.

**Client Comment** *Fixed and merged.*

```
20 event ClosingOnlySet(Symbol symbol, bool closingOnly);
```

## CVF-76. INFO

- **Category** Procedural
- **Source** StorageLib.sol

**Recommendation** Consider defining these values as hashes of library names for uniqueness.

**Client Comment** *Fixed and merged - not fully replaced because it’d need a re-eployment.*

```
24 uint256 private constant STORAGE_SLOT_BASE = 1_000_000;
```

```
120 uint256 private constant YIELD_STORAGE_SLOT_BASE = 2_000_000;
```

## CVF-77. FIXED

- **Category** Suboptimal
- **Source** StorageLib.sol

**Description** It is uncommon for a library to emit events.

**Recommendation** Consider emitting from contracts, that use this library.

**Client Comment** *Fixed and merged.*

```
88  emit FeeModelUpdated(symbol, feeModel);
```

```
93  emit ClosingOnlySet(symbol, closingOnly);
```

```
148 emit LadleSet(ladle);
```

```
157 emit CauldronSet(cauldron);
```

```
187 emit YieldInstrumentCreated(instrument, yieldInstrument);
```

```
275 emit TreasurySet(address(treasury));
```

```
284 emit PositionNFTSet(address(nft));
```

```
293 emit ClosingOnlySet(closingOnly);
```

```
302 emit TokenTrusted(token, trusted);
```

```
311 emit ProxyHashSet(proxyHash);
```

## CVF-78. FIXED

- **Category** Procedural
- **Source** StorageLib.sol

**Recommendation** This library should be defined in a separate file named “YieldStorageLib”.

**Client Comment** *Fixed and merged.*

```
115 library YieldStorageLib {
```

## CVF-79. FIXED

- **Category** Suboptimal
- **Source** StorageLib.sol

**Description** These functions contain more business logic, that it usually expected from a storage library.

**Recommendation** Consider moving them into a higher-level contract or library.

**Client Comment** *Fixed and merged.*

```
168 function createInstrument(Symbol symbol, bytes6 baseId, bytes6
    ↪ quoteId, IFeeModel feeModel)
```

```
190 function _createInstrument(
```

```
229 function _validInstrumentData(ICauldron cauldron, Symbol symbol,
    ↪ bytes6 baseId, bytes6 quoteId)
```

## CVF-80. FIXED

- **Category** Procedural
- **Source** StorageLib.sol

**Recommendation** This library should be defined in a separate file named "ConfigStorageLib".

**Client Comment** *Fixed and merged.*

```
256 library ConfigStorageLib {
```

## CVF-81. FIXED

- **Category** Procedural
- **Source** DataTypes.sol

**Description** We didn't review these files.

```
6 import {IFYToken} from "@yield-protocol/vault-v2/contracts/
    ↪ interfaces/IFYToken.sol";
import {IPool} from "@yield-protocol/yieldspace-tv/src/interfaces/
    ↪ IPool.sol";
```

## CVF-82. FIXED

- **Category** Flaw
- **Source** DataTypes.sol

**Recommendation** This calculation is incorrect. The actual number of bits occupied in the first slot is 224, so only 32 bits left.

**Client Comment** *Fixed and merged.*

```
33 //>slot0: 216bits used - 40bits left
```

## CVF-83. FIXED

- **Category** Procedural
- **Source** IContangoQuoter.sol

**Description** We didn't review these files.

```
4 import "../libraries/QuoterDataTypes.sol";
```

## CVF-84. INFO

- **Category** Suboptimal
- **Source** IContango.sol

**Recommendation** These functions should emit some events and these events should be declared in this interface.

**Client Comment** *Fixed and merged - events only duplicated on interface for doc purposes, can't re-use them because everything is in libraries and uni callback does not allow for returning data, therefore we don't the values to emit events afterwards.*

```
18 function createPosition(
```

```
37 function modifyPosition(
```

```
53 function modifyCollateral(
```

```
65 function deliver(PositionId positionId, address payer, address to)
     ↪ external payable;
```

ABDK
Consulting

## About us

Established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function.

The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

## Contact


**Email**

dmitry@abdkconsulting.com

**Website**

abdk.consulting

**Twitter**

twitter.com/ABDKconsulting

**LinkedIn**

linkedin.com/company/abdk-consulting