Report

v. 1.0

Customer

Reya Labs

Smart Contract Audit

Reya Network. Part III

4th April 2024

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	04.04.24	A. Zveryanskaya	Initial Draft
0.2	04.04.24	A. Zveryanskaya	Minor revision
1.0	04.04.24	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.

Reya Network is a trading-optimised modular L2 that focuses on three pillars of performance, liquidity and capital efficiency.

3 Project scope

We were asked to review the compared code with fixes for the Part I of the Audit:

core/		
CoreProxy.sol		
core/storage/		
Account.sol	AccountCollateral.sol	AccountRBAC.sol
AutoExchange Configuration.sol	BackstopLP Configuration.sol	Collateral Configuration.sol
CollateralPool.sol	Configuration Permissions.sol	Exchange.sol
GlobalCollateral Configuration.sol	IdStore.sol	Instrument Registrar.sol
InsuranceFund Configuration.sol	LimitConfiguration.sol	Liquidation Configuration.sol
Market.sol	Protocol Configuration.sol	RiskMatrix.sol
RiskMultipliers Configuration.sol		
core/modules/		
AccountModule.sol	AutoExchange ConfigurationModule.sol	CollateralModule.sol
CollateralPool Module.sol	ExecutionModule.sol	InstrumentModule.sol
InstrumentRegistrar Module.sol	InsuranceFund ConfigurationModule.sol	Protocol Configuration Module.sol
RiskConfiguration Module.sol		
core/modules/liquidation/		
Backstop LiquidationModule.sol	Common LiquidationModule.sol	RankedExecute LiquidationModule.sol
RankedSubmit LiquidationModule.sol		

core/modules/internal/		
MatchOrderModule.sol	DutchLiquidation Module.sol	
core/libraries/		
Permissions.sol	LiquidationBid PriorityQueue.sol	FeatureFlagSupport.sol
Events.sol	Errors.sol	DataTypes.sol
core/libraries/actions/		
CreateAccount.sol	EditCollateral.sol	
oracle-manager/modules/		
NodeModule.sol		
oracle-manager/nodes/		
ChainlinkNode.sol	ExternalNode.sol	
oracle-manager/storage/		
NodeDefinition.sol		

We have also reviewed the small scope addition:

rusd/utils/		
FeatureFlagSupport.sol		
utils/contracts/helpers/		
PrbMathHelper.sol		
utils/modules/modules/		
FeatureFlagModule.sol		
utils/modules/storage/		
FeatureFlag.sol		

Then we reviewed fixes for second part of the audit with corresponding files:

passive-pool/storage/		
ShareBalances.sol	PoolIdStore.sol	Pool.sol
GlobalConfiguration.sol		

passive-pool/modules/		
SharesModule.sol	ConfigurationModule.sol	
passive-pool/libraries/		
Events.sol	Errors.sol	
passive-pool/interfaces/		
ISharesModule.sol	IConfigurationModule.sol	
passive-perp/storage/		
TakerFee Configuration.sol	RebateConfiguration.sol	PerpPositions.sol
MarketConfiguration.sol	Market.sol	GlobalConfiguration.sol
passive-perp/modules/		
PassivePerpInstrument Module.sol	PassivePerpInformation Module.sol	ConfigurationModule.sol
passive-perp/libraries/		
QuadraticEquation.sol	PerpPositionSupport.sol	MaxExposure.sol
MatchOrders.sol	Liquidations.sol	FundingRate.sol
Events.sol	Errors.sol	DataTypes.sol
Constants.sol	IPassivePerp InformationModule.sol	

And the final part of the audit included the new files:

periphery/interfaces/external/		
ISocketController.sol	IConfigurationModule.sol	IDepositsModule.sol
ITransfersModule.sol	IWithdrawalsModule.sol	
periphery/libraries/		
Approvals.sol	CoreUtils.sol	DataTypes.sol
Deposits.sol	Errors.sol	Events.sol
Transfers.sol	Withdrawals.sol	
periphery/modules/		
ConfigurationModule.sol	DepositsModule.sol	TransfersModule.sol
WithdrawalsModule.sol		

periphery/storage/			
	Configuration.sol	GlobalConfiguration.sol	
rusd/interfaces/			
	IWrappedERC20.sol		
rusd/modules/			
	FeatureFlagModule.sol	OwnerUpgrade Module.sol	WrappedERC20 Module.sol
rusd/storage/			
	ERC20Storage.sol	WrapperStorage.sol	
rusd/utils/			
	ERC20.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 14 major, and a few less important issues. All identified Major issues have been fixed.

	Info	Fixed
Major	0	14
Moderate	3	3

Fixed 17 out of 20 issues

6 Major Issues

CVF-2 FIXED

- **Category** Suboptimal
- **Source** PassivePerpInformationModule.sol

Description The expression ”market.exists(marketId)” is calculated several times.

Recommendation Consider calculating once and reusing.

```
77 +Market.Data storage market = Market.exists(marketId);
```

```
79 +shortTrackers = Market.exists(marketId).getLatestTrackers(false,
      ↪ oraclePrice);
80 +longTrackers = Market.exists(marketId).getLatestTrackers(true,
      ↪ oraclePrice);
```

CVF-4 FIXED

- **Category** Suboptimal
- **Source** GlobalCollateralConfiguration.sol

Description In case ”assets” is zero, this function returns zero even is the collateral doesn’t exist.

Recommendation Consider always reverting on non-existing collateral to make problem investigation easier.

152
```
+if (assets == 0) {
+    return 0;
```

156
```
+exists(collateral);
```

175
```
+if (assets == 0) {
+    return 0;
```

179
```
+exists(collateral);
```

208
```
+if (shares == 0) {
+    return 0;
```

212
```
+exists(collateral);
```

231
```
+if (shares == 0) {
+    return 0;
```

239
```
+exists(collateral);
```

CVF-5 FIXED

- **Category** Suboptimal
- **Source** GlobalCollateralConfiguration.sol

Description This comment is ambiguous, as it is unclear, whether checked or unchecked overflow may happened.

Recommendation Consider using safe conversion to revert on overflow.

Client Comment *Introduced SafeCastU8 that checks max int8 against the value and reverts in case of overflow.*

```
314 +* - the collateral's␣decimals␣must␣be␣under␣max(int8),␣otherwise␣
        ↪ overflow␣will␣happen
    ␣␣␣␣␣␣
```

CVF-6 FIXED

- **Category** Suboptimal
- **Source** EditCollateral.sol

Description This argument is needed only when transferring tokens between margin accounts, as such transfer is implemented as withdrawal into the contract's balance and then depositing from it.

Recommendation Consider removing this argument and implementing a separate function for transferring tokens between accounts.

Client Comment *Replaced deposit and withdraw functions with 'addToAccount', 'tokenDeposit', 'tokenWithdraw' and 'deductFromAccount'. Esentially, when an in-between accounts transfer happens, we only use addToAccount and deductFromAccount. Then tokenDeposit and tokenWithdraw use msg.sender always, removing the need for depositFrom param.*

```
55 +address depositFrom
```

CVF-8 FIXED

- **Category** Suboptimal
- **Source** EditCollateral.sol

Description This check is needed only when transferring tokens between margin accounts, as such transfer is implemented as withdrawal into the contract's balance and then depositing from it.

Recommendation Consider removing this check and implementing a separate function for transferring tokens between accounts.

Client Comment *Replaced deposit and withdraw functions with 'addToAccount', 'tokenDeposit', 'tokenWithdraw' and 'deductFromAccount'. Esentially, when an in-between accounts transfer happens, we only use addToAccount and deductFromAccount. This check is removed.*

```
145 +if (withdrawTo != self) {
```

CVF-9 FIXED

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

Description Inserting a new constant into the middle of a enum constants list is a bad practice as this shifts values of other constants, so those who need to decode constant values off-chain would need to take the contract version into account.

Recommendation Consider always adding new constants into the end of a list.

Client Comment *Noted.*

```
21 +DutchLiquidation, // (core command) dutch liquidation of an account
```

CVF-10 FIXED

- **Category** Suboptimal
- **Source** Permissions.sol

Description Usually, permissions are identified by hashes of human-readable strings, rather than by plain strings. This also allows using permission names of arbitrary lengths.

Recommendation Consider using hashes instead of plain strings.

11
```
+bytes32 internal constant ADMIN = "ADMIN";
+bytes32 internal constant COLLATERAL_POOL_AUTOEXCHANGE_CONFIGURATOR
   ↪  = "CP_AUTO_EXCHANGE_CONFIGURATOR";
+bytes32 internal constant COLLATERAL_POOL_COLLATERAL_CONFIGURATOR =
   ↪  "CP_COLLATERAL_CONFIGURATOR";
+bytes32 internal constant COLLATERAL_POOL_LIMITS_CONFIGURATOR = "
   ↪ CP_LIMITS_CONFIGURATOR";
+bytes32 internal constant COLLATERAL_POOL_RISK_MATRIX_CONFIGURATOR
   ↪ = "CP_RISK_MATRIX_CONFIGURATOR";
+bytes32 internal constant
   ↪ COLLATERAL_POOL_INSURANCE_FUND_CONFIGURATOR = "
   ↪ CP_IF_CONFIGURATOR";
+bytes32 internal constant
   ↪ COLLATERAL_POOL_RISK_MULTIPLIERS_CONFIGURATOR = "
   ↪ CP_RISK_MULTIPLIERS_CONFIGURATOR";
+bytes32 internal constant COLLATERAL_POOL_LIQUIDATION_CONFIGURATOR
   ↪ = "CP_LIQUIDATION_CONFIGURATOR";
+bytes32 internal constant COLLATERAL_POOL_BACKSTOP_LP_CONFIGURATOR
   ↪ = "CP_BACKSTOP_LP_CONFIGURATOR";
```

CVF-11 FIXED

- **Category** Suboptimal
- **Source** ExecutionModule.sol

Description Executing transfer between account as withdraw plus deposit is suboptimal and overcomplicated.

Recommendation Consider implementing dedicated logic for such transfers and declare separate event for them.

Client Comment *Replaced deposit and withdraw functions with 'addToAccount', 'tokenDeposit', 'tokenWithdraw' and 'deductFromAccount'. This way we can reuse the functionality. The separate event is already declared, called TransferBetweenMarginAccounts.*

224
```
+account.withdraw(transferInput.collateral, transferInput.
   ↪ collateralAmount, address(this));
```

228
```
+destAccount.deposit(transferInput.collateral, transferInput.
   ↪ collateralAmount, address(this));
```

CVF-12 FIXED

- **Category** Suboptimal
- **Source** PrbMathHelper.sol

Description Flipping a number via multiplying it by minus one is suboptimal.

Recommendation Consider unwrapping, converting type, using unary minus and then wrapping.

95
```
+return a.intoSD59x18().mul(convert_sd(-1));
```

99
```
+return a.mul(convert_sd(-1));
```

CVF-13 FIXED

- **Category** Suboptimal
- **Source** ERC20.sol

Description Here allowance is updated in the storage before balance check is performed, which is inefficient in terms of gas.

Recommendation Consider refactoring to perform all necessary checks before updating the storage.

Client Comment *This was resolved by calling the '_transfer' function before subtracting from the allowance. Please confirm this does not introduce risks.*

```
162 store.allowance[from][msg.sender] -= amount;
```

CVF-14 FIXED

- **Category** Unclear behavior
- **Source** ERC20.sol

Description This allows changing the token name and symbol, while usually, these properties are assumed immutable and some tools and services may not be able to load the updated values.

Client Comment *The check above these lines ensures the initialization can only happen once, otherwise it reverts with 'revert InitError.AlreadyInitialized();'*

```
267 store.name = tokenName;
    store.symbol = tokenSymbol;
```

CVF-16 FIXED

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

Description This function only allows rescuing ether.

Recommendation Consider implementing some way to rescue tokens as well.

Client Comment *Implemented rescueErc20 function.*

```
79 function rescueFunds(uint256 amount) external override {
```

CVF-17 FIXED

- **Category** Suboptimal
- **Source** ConfigurationModule.sol

Description Use of the “transfer” function is discouraged.

Recommendation Consider using “send” instead.

Client Comment *Instead of transfer or send (which are limited by gas), we replaced it with ‘call{value:...}(“”)‘. PLease confirm no risks were added.*

```
83 payable(OwnableStorage.getOwner()).transfer(amount);
```

CVF-18 FIXED

- **Category** Suboptimal
- **Source** ConfigurationModule.sol

Description Always sending to the owner may be inconvenient.

Recommendation Consider passing the recipient address as an argument.

```
83 payable(OwnableStorage.getOwner()).transfer(amount);
```

7 Moderate Issues

CVF-19 FIXED

- **Category** Procedural
- **Source** SharesModule.sol

Description This typehash is not compliant with ERC-712.

Recommendation Consider making it compliant for compatibility with external tools and libraries.

Client Comment *Removed spaces. Also note that verifying chain id was moved from the domain to the typed data.*

```
26 +"RemoveLiquidityBySig(address␣caller,␣address␣owner,␣uint128␣poolId
   ↪ ,␣uint256␣sharesAmount,␣uint256␣minOut,␣uint256␣nonce,␣uint256
   ↪ ␣deadline)"
```

CVF-20 INFO

- **Category** Flaw
- **Source** EditCollateral.sol

Description Here, tokens could be transferred from the ”depositFrom” address without explicit consent of the owner of this address, which is potentially very dangerous. The only safe value for ”depositFrom” here is ”msg.sender”.

Recommendation Consider explicitly requiring ”depositFrom” to equal ”msg.sender” here.

Client Comment *This is an internal function and the depositFrom can only be msg.sender or the contract itself based on the entire flow.*

```
81 +collateral.safeTransferFrom(depositFrom, self, collateralAmount);
```

CVF-21 INFO

- **Category** Flaw
- **Source** EditCollateral.sol

Description In case depositFrom == self, no event is emitted, while balance is still updated.

Recommendation Consider emitting event each time some balance is updated.

Client Comment *An event is already emitted by 'updateBalance'. Otherwise, when depositFrom == self then it means this is a transfer between two margin accounts and this is signaled by the 'TransferBetweenMarginAccounts' event.*

```
85 +    emit Events.Deposited(account.id, collateral, collateralAmount,
   ↪  depositFrom, block.timestamp);
```

```
90 +AccountCollateral.updateBalance(account, collateral,
   ↪ collateralAmount.toInt());
```

CVF-22 INFO

- **Category** Flaw
- **Source** EditCollateral.sol

Description In case withdrawTo == self, no event is emitted, while balance is still updated.

Recommendation Consider emitting event each time some balance is updated.

Client Comment *An event is already emitted by 'updateBalance'. Otherwise, when withdrawTo == self then it means this is a transfer between two margin accounts and this is signaled by the 'TransferBetweenMarginAccounts' event.*

```
135 +AccountCollateral.updateBalance(account, collateral, -
   ↪ collateralAmount.toInt());
```

```
147 +    emit Events.Withdrawn(account.id, collateral, collateralAmount,
   ↪  withdrawTo, block.timestamp);
```

CVF-23 FIXED

- **Category** Procedural
- **Source** ExecutionModule.sol

Recommendation This typehash is not compliant with ERC-712. Should be: ExecuteBySig(address caller,uint128 accountId,Command[] commands,uint256 nonce,uint256 deadline)Command(uint256 commandType,bytes inputs,uint128 marketId,uint128 exchangeId)

Client Comment *Fixed – removed spaces and defined the Command type. Also note that verifying chain id was moved from the domain to the typed data.*

```
61 +"ExecuteBySig(address␣caller,␣uint128␣accountId,␣(uint256,␣bytes,␣
      ↪ uint128,␣uint128)[]␣commands,␣uint256␣nonce,␣uint256␣deadline)
      ↪ "
```

CVF-24 FIXED

- **Category** Procedural
- **Source** ExecutionModule.sol

Description This hashing format is not compliant with ERC-712.

Recommendation Consider making is compliant for compatibility with existing tools and libraries.

Client Comment *Fixed – new helper functions were created and added in SignatureHelper.sol. The command is first hashed before encoded into ExecuteBySig.*

```
109 +keccak256(abi.encode(EXECUTE_TYPYHASH, msg.sender, accountId,
       ↪ commands, incrementedNonce, sig.deadline))
```

8 Recommendations

CVF-3 INFO

- **Category** Procedural
- **Source** GlobalCollateralConfiguration.sol

Description In ERC20 the ”decimals” property is optimal and is used by UI to render token amounts in a human-readable way. Using this property in smart contracts is discouraged.

Recommendation Consider treating all token amounts as integers.

Client Comment *The community multisig controls what tokens are deployed and used by the protocol. This will ensure that all of them implement the decimals() correctly.*

```
103 +collateralDecimals = IERC20(collateralAddress).decimals();
```

CVF-25 INFO

- **Category** Procedural
- **Source** ConfigurationModule.sol

Description We didn’t review this file.

```
10 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
     ↪ helpers/CallChecker.sol";
```

CVF-26 INFO

- **Category** Procedural
- **Source** SharesModule.sol

Description We didn’t review this file.

```
10 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
     ↪ helpers/CallChecker.sol";
```

CVF-27 FIXED

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

Recommendation The ”quoteToken” and ”accountId” parameters should be indexed.

```
27 +event PoolCreated(uint128 indexed poolId, address quoteToken,
      ↪ uint128 accountId, uint256 blockTimestamp);
```

CVF-28 INFO

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

Recommendation The type for the ”quoteToken” parameter should be ”IERC20”.

Client Comment *For consistency, we prefer to keep it as address.*

```
27 +event PoolCreated(uint128 indexed poolId, address quoteToken,
      ↪ uint128 accountId, uint256 blockTimestamp);
```

CVF-30 INFO

- **Category** Procedural
- **Source** Market.sol

Description We didn’t review this file.

```
57 +import { SD1x18 } from "@prb/math/SD1x18.sol";
```

CVF-31 INFO

- **Category** Procedural
- **Source** ConfigurationModule.sol

Description We didn’t review this file.

```
10 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
      ↪ helpers/CallChecker.sol";
```

CVF-32 INFO

- **Category** Procedural
- **Source** PassivePerpInstrumentModule.sol

Recommendation This could be simplified as "^0.8.19". Also relevant for: ConfigurationPermissions.sol, DutchLiquidationModule.sol, MatchOrderModule.sol, FeatureFlagSupport.sol, ERC20.sol, WrapperStorage.sol, ERC20Storage.sol, FeatureFlagModule.sol, OwnerUpgradeModule.sol, WrappedERC20Module.sol, IWrappedERC20.sol, GlobalConfiguration.sol, Configuration.sol, ConfigurationModule.sol, DepositsModule.sol, TransfersModule.sol, WithdrawalsModule.sol, Events.sol, DataTypes.sol, Approvals.sol, CoreUtils.sol, Errors.sol, Deposits.sol, Withdrawals.sol, Transfers.sol, ISocketController.sol, IConfigurationModule.sol, IDepositsModule.sol, ITransfersModule.sol, IWithdrawalsModule.sol.

Client Comment *Consistent with previous audits.*

```
9 +pragma solidity >=0.8.19 <0.9.0;
```

CVF-33 INFO

- **Category** Procedural
- **Source** PassivePerpInstrumentModule.sol

Description We didn't review this file.

```
11 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
     ↪ helpers/CallChecker.sol";
```

CVF-34 INFO

- **Category** Suboptimal
- **Source** PassivePerpInstrumentModule.sol

Description The expression "market.getConfig()" is calculated twice.

Recommendation Consider calculating once and reusing.

Client Comment *Inefficiency introduced to avoid "stack too deep" compilation error.*

```
148 +minimumOrderBase: market.getConfig().minimumOrderBase,
    +baseSpacing: market.getConfig().baseSpacing,
```

CVF-35 FIXED

- **Category** Suboptimal
- **Source** Errors.sol

Recommendation This error could be made more helpful by including the invalid liquidation type as a parameter.

```
39 +error InvalidLiquidationType();
```

CVF-37 INFO

- **Category** Procedural
- **Source** MaxExposure.sol

Description We didn't review this file.

```
14 +import { SD1x18 } from "@prb/math/SD1x18.sol";
```

CVF-39 INFO

- **Category** Procedural
- **Source** CollateralConfiguration.sol

Description We didn't review this file.

```
30 +import { UNIT, ud } from "@prb/math/UD60x18.sol";
```

CVF-40 FIXED

- **Category** Suboptimal
- **Source** ProtocolConfiguration.sol

Recommendation The "_STORAGE_NAME" value is already a hash, no reason to hash it again.

```
51 +bytes32 s = keccak256(abi.encodePacked(_STORAGE_NAME));
```

CVF-41 INFO

- **Category** Procedural
- **Source** GlobalCollateralConfiguration.sol

Description We didn't review this file.

```
19 +import { DecimalMath } from "@voltz-protocol/util-contracts/src/
      ↪ helpers/DecimalMath.sol";
```

CVF-42 INFO

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

Recommendation The type for this field should be "IERC20".

Client Comment *For consistency, we prefer to keep it as address.*

```
40 +address[] collaterals;
```

CVF-43 INFO

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

Recommendation The key type for these mappings should be "IERC20".

Client Comment *For consistency, we prefer to keep it as address.*

```
41 +mapping(address collateral => GlobalCollateralConfig) configs;
   +mapping(address collateral => GlobalCachedCollateralConfig)
      ↪ cachedConfigs;
```

CVF-44 INFO

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

Recommendation The type for collateral arguments should be ”IERC20”.

Client Comment *For consistency, we prefer to keep it as address.*

```
57 +function set(address collateralAddress, GlobalCollateralConfig
   ↪ memory config) internal {
```

```
126 +function exists(address collateral) internal view {
```

```
151 +function convertToShares(address collateral, uint256 assets)
   ↪ internal view returns (uint256) {
```

```
174 +function convertToShares(address collateral, int256 assets)
   ↪ internal view returns (int256) {
```

```
207 +function convertToAssets(address collateral, uint256 shares)
   ↪ internal view returns (uint256) {
```

```
230 +function convertToAssets(address collateral, int256 shares)
   ↪ internal view returns (int256) {
```

```
275 +function checkWithdrawLimits(address collateralAddress, uint256
   ↪ assets) internal {
```

CVF-45 INFO

- **Category** Procedural
- **Source** ConfigurationPermissions.sol

Description We didn’t review this file.

```
12 +import { SetUtil } from "@voltz-protocol/util-contracts/src/helpers
   ↪ /SetUtil.sol";
```

CVF-46 INFO

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

Description This event is emitted even if nothing actually changed.

Client Comment *Not concerned.*

```
49 +emit Events.ConfigurationPermissionRevoked({
```

CVF-47 INFO

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

Recommendation The return type should be ”IERC20”.

Client Comment *For consistency, we prefer to keep it as address.*

```
443 +function getParent(Data storage self, address collateral) internal
    ↪ view returns (address parentCollateral) {
```

CVF-48 INFO

- **Category** Procedural
- **Source** Account.sol

Description We didn’t review this file.

```
25 +import { SetUtil } from "@voltz-protocol/util-contracts/src/helpers
    ↪ /SetUtil.sol";
```

CVF-49 INFO

- **Category** Procedural
- **Source** RiskMatrix.sol

Description We didn’t review this file.

```
18 +import { SD1x18 } from "@prb/math/SD1x18.sol";
```

CVF-50 INFO

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

Recommendation The type for this argument should be ”IERC20”.

Client Comment *For consistency, we prefer to keep it as address.*

```
53 +address collateral,
```

CVF-51 FIXED

- **Category** Documentation
- **Source** EditCollateral.sol

Description Unlike other argument, this argument isn’t documented.

Recommendation Consider documenting.

```
55 +address depositFrom
```

CVF-52 INFO

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

Recommendation The type for this field should be ”IERC20”.

Client Comment *For consistency, we prefer to keep it as address.*

```
161 +address quoteCollateral;
```

```
557 +address collateral;
```

CVF-53 INFO

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

Recommendation The type for the collateral parameters should be ”IERC20”.

Client Comment *For consistency, we prefer to keep it as address.*

```
128 +event QuoteCollateralsUpdated(uint128 indexed collateralPoolId,
    ↪ address[] quoteCollaterals, uint256 blockTimestamp);
```

```
138 +    address indexed quoteCollateral, uint256[]
    ↪ blocksOfQuoteCollateral, uint256 blockTimestamp
```

```
171 +    address indexed collateralAddress,
```

```
311 +    address indexed collateral,
```

CVF-54 INFO

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

Recommendation The type for the instrument parameters should be more specific.

Client Comment *For consistency, we prefer to keep it as address.*

```
184 +event InstrumentRegistrationUpdated(address indexed
    ↪ instrumentAddress, bool isRegistered);
```

```
219 +event InstrumentRegistrationFlagSet(address indexed
    ↪ instrumentAddress, bool isRegistered, uint256 blockTimestamp);
```

CVF-55 INFO

- **Category** Procedural
- **Source** Errors.sol

Description We didn’t review this file.

```
13 +import { SD1x18 } from "@prb/math/SD1x18.sol";
```

CVF-56 INFO

- **Category** Procedural
- **Source** DutchLiquidationModule.sol

Description We didn't review this file.

```
10 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
     ↪ helpers/CallChecker.sol";
```

CVF-57 INFO

- **Category** Procedural
- **Source** MatchOrderModule.sol

Description We didn't review this file.

```
15 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
     ↪ helpers/CallChecker.sol";
```

CVF-58 INFO

- **Category** Suboptimal
- **Source** MatchOrderModule.sol

Description The expression "Exchange.exists(exchangeId)" is calculated twice.

Recommendation Consider calculating once and reusing.

Client Comment *If we save Exchange.exists(exchangeId) and then calculate the fees, we risk a revert. Other options (like try-catch) feel too complex.*

```
54 +bool creditExchangeFees = exchangeId != 0 && Exchange.exists(
     ↪ exchangeId).ownsPass();
```

```
95 +    Exchange.Data storage exchange = Exchange.exists(exchangeId);
```

CVF-59 FIXED

- **Category** Procedural
- **Source** MatchOrderModule.sol

Recommendation This check should be done earlier.

```
56 +if (counterpartyAccountIds.length == 0) {
```

CVF-60 INFO

- **Category** Procedural
- **Source** RankedExecuteLiquidationModule.sol

Description We didn't review this file.

```
11 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
   ↪ helpers/CallChecker.sol";
```

CVF-61 INFO

- **Category** Procedural
- **Source** CommonLiquidationModule.sol

Description We didn't review this file.

```
11 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
   ↪ helpers/CallChecker.sol";
```

CVF-62 INFO

- **Category** Procedural
- **Source** BackstopLiquidationModule.sol

Description We didn't review this file.

```
11 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
   ↪ helpers/CallChecker.sol";
```

CVF-63 INFO

- **Category** Procedural
- **Source** RankedSubmitLiquidationModule.sol

Description We didn't review this file.

```
11 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
   ↪ helpers/CallChecker.sol";
```

CVF-64 INFO

- **Category** Procedural
- **Source** AutoExchangeConfiguration-Module.sol

Description We didn't review this file.

```
11 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
   ↪ helpers/CallChecker.sol";
```

CVF-65 INFO

- **Category** Procedural
- **Source** AccountModule.sol

Description We didn't review these files.

```
11 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
   ↪ helpers/CallChecker.sol";
```

```
34 +import { SD1x18 } from "@prb/math/SD1x18.sol";
```

CVF-66 INFO

- **Category** Procedural
- **Source** CollateralModule.sol

Description We didn't review this file.

```
11 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
   ↪ helpers/CallChecker.sol";
```

CVF-67 INFO

- **Category** Procedural
- **Source** CollateralPoolModule.sol

Description We didn't review this file.

```
11 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
     ↪ helpers/CallChecker.sol";
```

CVF-68 INFO

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

Recommendation The type for this argument should be "IERC20".

Client Comment *For consistency, we prefer to keep it as address.*

```
88 +address collateral
```

CVF-69 INFO

- **Category** Procedural
- **Source** ExecutionModule.sol

Description We didn't review this file.

```
11 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
     ↪ helpers/CallChecker.sol";
```

CVF-70 INFO

- **Category** Procedural
- **Source** InstrumentModule.sol

Description We didn't review this file.

```
11 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
     ↪ helpers/CallChecker.sol";
```

CVF-71 INFO

- **Category** Procedural
- **Source** InstrumentRegistrarModule.sol

Description We didn't review this file.

```
11 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
   ↪ helpers/CallChecker.sol";
```

CVF-72 INFO

- **Category** Procedural
- **Source** InsuranceFundConfigura-tionModule.sol

Description We didn't review this file.

```
11 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
   ↪ helpers/CallChecker.sol";
```

CVF-73 INFO

- **Category** Procedural
- **Source** ProtocolConfigurationModule.sol

Description We didn't review this file.

```
11 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
   ↪ helpers/CallChecker.sol";
```

CVF-74 INFO

- **Category** Procedural
- **Source** RiskConfigurationModule.sol

Description We didn't review these files.

```
11 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
      ↪ helpers/CallChecker.sol";
```

```
26 +import { SD1x18 } from "@prb/math/SD1x18.sol";
```

CVF-75 INFO

- **Category** Procedural
- **Source** FeatureFlag.sol

Description We didn't review these files.

```
5  +import { SetUtil } from "@voltz-protocol/util-contracts/src/helpers
      ↪ /SetUtil.sol";
   +import { OwnableStorage } from "@voltz-protocol/util-contracts/src/
      ↪ ownership/Ownable.sol";
```

CVF-76 INFO

- **Category** Procedural
- **Source** FeatureFlagModule.sol

Description We didn't review this file.

```
5  +import { CallChecker } from "@voltz-protocol/util-contracts/src/
      ↪ helpers/CallChecker.sol";
```

CVF-77 INFO

- **Category** Procedural
- **Source** NodeModule.sol

Description We didn't review this file.

```
11 +import { CallChecker } from "@voltz-protocol/util-contracts/src/
      ↪ helpers/CallChecker.sol";
```

CVF-79 INFO

- **Category** Procedural
- **Source** ERC20.sol

Description We didn't review these files.

```
13 import { CallChecker } from "@voltz-protocol/util-contracts/src/
      ↪ helpers/CallChecker.sol";
   import { IERC20 } from "@voltz-protocol/util-contracts/src/
      ↪ interfaces/IERC20.sol";
   import { InitError } from "@voltz-protocol/util-contracts/src/errors
      ↪ /InitError.sol";
   import { ParameterError } from "@voltz-protocol/util-contracts/src/
      ↪ errors/ParameterError.sol";
```

CVF-80 INFO

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

Recommendation The type for this field should be "IERC20".

Client Comment *For consistency, we prefer to keep it as address.*

```
22 address underlyingAsset;
```

CVF-81 INFO

- **Category** Documentation
- **Source** ERC20Storage.sol

Description The order of the keys is unclear.

Recommendation Consider giving descriptive named to the keys and/or documenting.

Client Comment *This is very common functionality. Adding names would bulk the code.*

```
22 mapping(address => mapping(address => uint256)) allowance;
```

CVF-83 INFO

- **Category** Procedural
- **Source** FeatureFlagModule.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
19 contract FeatureFlagModule is BaseFeatureFlagModule { }
```

CVF-85 INFO

- **Category** Procedural
- **Source** OwnerUpgradeModule.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
14 contract OwnerUpgradeModule is BaseOwnerUpgradeModule { }
```

CVF-86 INFO

- **Category** Procedural
- **Source** WrappedERC20Module.sol

Description We didn't review these files.

```
10 import { CallChecker } from "@voltz-protocol/util-contracts/src/
   ↪ helpers/CallChecker.sol";
```

```
17 import { IERC20 } from "@voltz-protocol/util-contracts/src/
   ↪ interfaces/IERC20.sol";
   import { ERC20Helper } from "@voltz-protocol/util-contracts/src/
   ↪ token/ERC20Helper.sol";
   import { OwnableStorage } from "@voltz-protocol/util-contracts/src/
   ↪ storage/OwnableStorage.sol";
```

CVF-87 INFO

- **Category** Bad datatype
- **Source** WrappedERC20Module.sol

Recommendation The type for this argument should be “IERC20”.

Client Comment *For consistency, we prefer to keep it as address.*

```
28 address underlyingAsset
```

CVF-88 FIXED

- **Category** Suboptimal
- **Source** WrappedERC20Module.sol

Recommendation This variable is redundant, as “msg.sender” is cheaper to access than a local variable.

```
87 address from = msg.sender;
```

CVF-89 INFO

- **Category** Procedural
- **Source** IWrappedERC20.sol

Description We didn’t review this file.

```
10 import { IERC20 } from "@voltz-protocol/util-contracts/src/
   ↪ interfaces/IERC20.sol";
```

CVF-90 INFO

- **Category** Bad datatype
- **Source** IWrappedERC20.sol

Recommendation The type for this argument should be “IERC20”.

Client Comment *For consistency, we prefer to keep it as address.*

```
25 address underlyingAsset
```

CVF-91 FIXED

- **Category** Unclear behavior
- **Source** IWrappedERC20.sol

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

Client Comment *Added event DepositRUSD(address indexed from, uint256 amount, address indexed to); event WithdrwaRUSD(address indexed from, uint256 amount, address indexed to);*

```
32 function deposit(uint256 amount) external;
```

```
38 function withdraw(uint256 amount) external;
```

```
45 function depositTo(uint256 amount, address to) external;
```

```
52 function withdrawTo(uint256 amount, address to) external;
```

CVF-92 INFO

- **Category** Bad datatype
- **Source** IWrappedERC20.sol

Recommendation The return type should be “IERC20”.

Client Comment *For consistency, we prefer to keep it as address.*

```
58 function getUnderlyingAsset() external view returns (address);
```

CVF-94 INFO

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

Recommendation The type for these fields should be more specific.

Client Comment *For consistency, we prefer to keep it as address.*

```
21 address coreProxy;
```

```
25 address rUSDProxy;
```

```
29 address passivePoolProxy;
```

CVF-95 INFO

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

Recommendation The return type should be more specific.

Client Comment *For consistency, we prefer to keep it as address.*

```
59 function getUSDCAddress(address rUSDProxy) internal view returns (
   ↪ address) {
```

```
66 function getCoreProxyAddress() internal view returns (address) {
```

```
73 function getRUSDProxyAddress() internal view returns (address) {
```

```
80 function getPassivePoolProxyAddress() internal view returns (address
   ↪ ) {
```

CVF-96 INFO

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

Recommendation The argument type should be ”IWrappedERC20”.

Client Comment *For consistency, we prefer to keep it as address.*

```
59 function getUSDCAddress(address rUSDProxy) internal view returns (
   ↪ address) {
```

CVF-97 INFO

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

Recommendation The ”token” key type for this mapping should be ”IERC20”.

Client Comment *For consistency, we prefer to keep it as address.*

```
22 mapping(address token => address controller) tokenControllers;
```

```
26 mapping(address token => mapping(address connector => uint256 price)
   ↪ ) staticWithdrawFee;
```

CVF-98 INFO

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

Recommendation The ”controller” value type should be more specific.

Client Comment *For consistency and because we don’t directly control the interface, we prefer to keep it as address.*

```
22 mapping(address token => address controller) tokenControllers;
```

CVF-99 INFO

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

Recommendation The ”connector” key type should be more specific.

Client Comment *For consistency and because we don’t directly control the interface, we prefer to keep it as address.*

```
26 mapping(address token => mapping(address connector => uint256 price)
   ↪ ) staticWithdrawFee;
```

CVF-100 INFO

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

Recommendation The ”token” argument type should be ”IERC20”.

Client Comment *For consistency, we prefer to keep it as address.*

```
41 function setTokenController(address token, address controller)
   ↪ internal {
```

```
52 function setTokenStaticWithdrawFee(address token, address connector,
   ↪  uint256 fee) internal {
```

```
63 function ensureControllerIsRegistered(address token, address
   ↪ controller) internal view {
```

```
78 function getController(address token) internal view returns (address
   ↪  controller) {
```

```
91 function getStaticWithdrawFee(address token, address connector)
   ↪ internal view returns (uint256 price) {
```

CVF-101 INFO

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

Recommendation The ”controller” argument type should be more specific.

Client Comment *For consistency and because we don't directly control the interface, we prefer to keep it as address.*

```
41 function setTokenController(address token, address controller)
   ↪ internal {
```

```
63 function ensureControllerIsRegistered(address token, address
   ↪ controller) internal view {
```

CVF-102 INFO

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

Recommendation The ”connector” argument type should be more specific.

Client Comment *For consistency and because we don't directly control the interface, we prefer to keep it as address.*

```
52 function setTokenStaticWithdrawFee(address token, address connector,
   ↪  uint256 fee) internal {
```

```
91 function getStaticWithdrawFee(address token, address connector)
   ↪ internal view returns (uint256 price) {
```

CVF-103 INFO

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

Recommendation The return type should be more specific.

Client Comment *For consistency and because we don't directly control the interface, we prefer to keep it as address.*

```
78 function getController(address token) internal view returns (address
   ↪  controller) {
```

CVF-104 INFO

- **Category** Procedural
- **Source** ConfigurationModule.sol

Description We didn't review these files.

```
13 import { CallChecker } from "@voltz-protocol/util-contracts/src/
   ↪ helpers/CallChecker.sol";
   import { OwnableStorage } from "@voltz-protocol/util-contracts/src/
   ↪ storage/OwnableStorage.sol";
```

CVF-105 INFO

- **Category** Procedural
- **Source** DepositsModule.sol

Description We didn't review this file.

```
14 import { CallChecker } from "@voltz-protocol/util-contracts/src/
   ↪ helpers/CallChecker.sol";
```

CVF-106 INFO

- **Category** Procedural
- **Source** TransfersModule.sol

Description We didn't review this file.

```
13 import { CallChecker } from "@voltz-protocol/util-contracts/src/
   ↪ helpers/CallChecker.sol";
```

CVF-107 INFO

- **Category** Procedural
- **Source** WithdrawalsModule.sol

Description We didn't review this file.

```
13 import { CallChecker } from "@voltz-protocol/util-contracts/src/
   ↪ helpers/CallChecker.sol";
```

CVF-108 INFO

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

Recommendation Events are usually named via nouns, such as ”GlobalConfiguration” or ”Controller”.

```
21 event GlobalConfigurationUpdated(GlobalConfiguration.Data
   ↪ globalConfig, uint256 blockTimestamp);

29 event ControllerUpdated(address token, address controller, uint256
   ↪ blockTimestamp);

37 event FeeUpdated(address token, uint256 fee, uint256 blockTimestamp)
   ↪ ;
```

CVF-109 INFO

- **Category** Suboptimal
- **Source** Events.sol

Recommendation The ”blockTimestamp” parameters are redundant, as every emitted even is anyway bound to a block that has a timestamp.

Client Comment *Same as in previous audits, we use this param to avoid double RPC query.*

```
21 event GlobalConfigurationUpdated(GlobalConfiguration.Data
   ↪ globalConfig, uint256 blockTimestamp);

29 event ControllerUpdated(address token, address controller, uint256
   ↪ blockTimestamp);

37 event FeeUpdated(address token, uint256 fee, uint256 blockTimestamp)
   ↪ ;
```

CVF-110 INFO

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

Recommendation The type for the ”token” parameters should be ”IERC20”.

Client Comment *For consistency, we prefer to keep it as address.*

```
29 event ControllerUpdated(address token, address controller, uint256
   ↪ blockTimestamp);
```

```
37 event FeeUpdated(address token, uint256 fee, uint256 blockTimestamp)
   ↪ ;
```

CVF-111 FIXED

- **Category** Suboptimal
- **Source** Events.sol

Recommendation The ”token” and ”controller” parameters should be indexed.

```
29 event ControllerUpdated(address token, address controller, uint256
   ↪ blockTimestamp);
```

```
37 event FeeUpdated(address token, uint256 fee, uint256 blockTimestamp)
   ↪ ;
```

CVF-113 INFO

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

Recommendation The type for this field should be ”IERC20”.

Client Comment *For consistency, we prefer to keep it as address.*

```
20  address token;
```

```
36  address socketConnector;
```

```
67  address socketConnector;
```

```
82  address socketConnector;
```

```
112 address token;
```

```
127 address token;
```

CVF-114 INFO

- **Category** Procedural
- **Source** Approvals.sol

Description We didn’t review this file.

```
10  import { ERC20Helper } from "@voltz-protocol/util-contracts/src/
    ↪ token/ERC20Helper.sol";
```

CVF-115 INFO

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

Recommendation The type for the ”erc20Token” argument should be ”IERC20”.

Client Comment *For consistency, we prefer to keep it as address.*

```
21  function executeNewApproval(address erc20Token, address spender,
    ↪ uint256 tokenAmount) internal {
```

CVF-117 INFO

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

Recommendation The type for the ”token” arguments should be ”IERC20”.

Client Comment *For consistency, we prefer to keep it as address.*

```
25 function coreWithdraw(uint128 accountId, address token, uint256
   ↪ tokenAmount, EIP712Signature memory sig) internal {
```

```
38     address token,
```

```
59 function coreDeposit(uint128 accountId, address token, uint256
   ↪ tokenAmount) internal {
```

```
76     address token,
```

CVF-118 INFO

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

Recommendation The type for the ”token” parameter should be ”IERC20”.

Client Comment *For consistency, we prefer to keep it as address.*

```
18 error ControllerNotRegistered(address token, address controler);
```

CVF-119 INFO

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

Recommendation The type for the ”controller” parameter should be more specific.

Client Comment *For consistency and because we don’t directly control the interface, we prefer to keep it as address.*

```
18 error ControllerNotRegistered(address token, address controler);
```

CVF-120 FIXED

- **Category** Readability
- **Source** Errors.sol

Recommendation Typo in ”controler” parameter name.

```
18 error ControllerNotRegistered(address token, address controler);
```

CVF-121 FIXED

- **Category** Suboptimal
- **Source** Errors.sol

Recommendation This error could be made more useful by adding certain parameters into it.

Client Comment *Added caller and account owner as params.*

```
23 error UnauthorizedCaller();
```

CVF-122 INFO

- **Category** Procedural
- **Source** Deposits.sol

Description We didn’t review this file.

```
18 "@voltz-protocol/util-modules/src/interfaces/
     ↪ IBaseAssociatedSystemsModule.sol";
```

CVF-123 INFO

- **Category** Procedural
- **Source** Withdrawals.sol

Description We didn’t review this file.

```
17 import { BridgingUtils } from "./BridgingUtils.sol";
```

CVF-124 INFO

- **Category** Procedural
- **Source** Transfers.sol

Description We didn't review this file.

```
14 import { IConfigurationModule } from "@voltz-protocol/passive-pool/
   ↪ src/interfaces/IConfigurationModule.sol";
```

CVF-125 INFO

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

Recommendation The type for the "token" arguments should be "IERC20".

Client Comment *For consistency, we prefer to keep it as address.*

```
33 function setTokenController(address token, address controller)
   ↪ external;
```

```
40 function getTokenController(address token) external view returns (
   ↪ address controller);
```

```
48 function setTokenStaticWithdrawFee(address token, address connector,
   ↪  uint256 staticFee) external;
```

```
56 function getTokenStaticWithdrawFee(address token, address connector)
   ↪  external view returns (uint256 staticFee);
```

CVF-126 INFO

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

Recommendation The type for the "controller" argument should be more specific.

Client Comment *For consistency and because we don't directly control the interface, we prefer to keep it as address.*

```
33 function setTokenController(address token, address controller)
   ↪ external;
```

CVF-127 INFO

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

Recommendation The return type should be more specific.

Client Comment *For consistency and because we don't directly control the interface, we prefer to keep it as address.*

```
40 function getTokenController(address token) external view returns (
      ↪ address controller);
```

CVF-128 INFO

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

Recommendation The type for the "connector" arguments should be more specific.

Client Comment *For consistency and because we don't directly control the interface, we prefer to keep it as address.*

```
48 function setTokenStaticWithdrawFee(address token, address connector,
      ↪  uint256 staticFee) external;
```

```
56 function getTokenStaticWithdrawFee(address token, address connector)
      ↪  external view returns (uint256 staticFee);
```

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