CODESPECT

CoreWriter

SECURITY ASSESSMENT REPORT

August 25, 2025

Prepared for Hyperwave

Contents

1 About CODESPECT 2

2 Disclaimer 2

3 Risk Classification 3

4 Executive Summary 4

5 Audit Summary 5
- 5.1 Scope - Audited Files . . . 5
- 5.2 Findings Overview . . . 6
- 5.3 Fix Review Phase - Findings Overview . . . 6

6 System Overview 7
- 6.1 Configurator . . . 7
- 6.2 HyperCoreAccount . . . 7
- 6.3 Managers . . . 7
  - 6.3.1 TradeStakeManager . . . 7
  - 6.3.2 VaultManager . . . 8
- 6.4 System Flow . . . 8

7 Issues 9
- 7.1 [Low] Blocked whitelist removals for de-listed accounts . . . 9
- 7.2 [Low] Joint withdrawal and spot transfer whitelist could lead to locked funds . . . 9
- 7.3 [Low] Receive of Native token is enabled while withdrawal is not possible . . . 9
- 7.4 [Info] Introduce validation of the order size . . . 10
- 7.5 [Info] Replace storage writes with memory ones . . . 10

8 Fix Review Phase Issues 11
- 8.1 [Info] Possible overflow during price tolerance control . . . 11

9 Centralisation Risks 12

10 Evaluation of Provided Documentation 13

11 Test Suite Evaluation 14
- 11.1 Compilation Output . . . 14
- 11.2 Tests Output . . . 15
- 11.3 Notes on the Test Suite . . . 23

1 About CODESPECT

CODESPECT is a specialized smart contract security firm dedicated to ensure the safety, reliability, and success of blockchain projects. Our services include comprehensive smart contract audits, secure design and architecture consultancy, and smart contract development across leading blockchain platforms such as Ethereum (Solidity), Starknet (Cairo), and Solana (Rust).

At CODESPECT, we are committed to build secure, resilient blockchain infrastructures. We provide strategic guidance and technical expertise, working closely with our partners from concept development through deployment. Our team consists of blockchain security experts and seasoned engineers who apply the latest auditing and security methodologies to help prevent exploits and vulnerabilities in your smart contracts.

Smart Contract Auditing: Security is at the core of everything we do at CODESPECT. Our auditors conduct thorough security assessments of smart contracts written in Solidity, Cairo, and Rust, ensuring that they function as intended without vulnerabilities. We specialize in providing tailored security solutions for projects on EVM-compatible chains and Starknet. Our audit process is highly collaborative, keeping clients involved every step of the way to ensure transparency and security. Our team is also dedicated to cutting-edge research, ensuring that we stay ahead of emerging threats.

Secure Design & Architecture Consultancy: At CODESPECT, we believe that secure development begins at the design phase. Our consultancy services offer deep insights into secure smart contract architecture and blockchain system design, helping you build robust, secure, and scalable decentralized applications. Whether you're working with Ethereum, Starknet, or other blockchain platforms, our team helps you navigate the complexity of blockchain development with confidence.

Tailored Cybersecurity Solutions: CODESPECT offers specialized cybersecurity solutions designed to minimize risks associated with traditional attack vectors, such as phishing, social engineering, and Web2 vulnerabilities. Our solutions are crafted to address the unique security needs of blockchain-based applications, reducing exposure to attacks and ensuring that all aspects of the system are fortified.

With a focus on the intersection of security and innovation, CODESPECT strives to be a trusted partner for blockchain projects at every stage of development and for each aspect of security.

2 Disclaimer

Limitations of this Audit: This report is based solely on the materials and documentation provided to CODESPECT for the specific purpose of conducting the security review outlined in the Summary of Audit and Files. The findings presented in this report may not be comprehensive and may not identify all possible vulnerabilities. CODESPECT provides this review and report on an "as-is" and "as-available" basis. You acknowledge that your use of this report, including any associated services, products, protocols, platforms, content, and materials, is entirely at your own risk.

Inherent Risks of Blockchain Technology: Blockchain technology is still evolving and is inherently subject to unknown risks and vulnerabilities. This review focuses exclusively on the smart contract code provided and does not cover the compiler layer, underlying programming language elements beyond the reviewed code, or any other potential security risks that may exist outside of the code itself.

Purpose and Reliance of this Report: This report should not be viewed as an endorsement of any specific project or team, nor does it guarantee the absolute security of the audited smart contracts. Third parties should not rely on this report for any purpose, including making decisions related to investments or purchases.

Liability Disclaimer: To the maximum extent permitted by law, CODESPECT disclaims all liability for the contents of this report and any related services or products that arise from your use of it. This includes but is not limited to, implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Third-Party Products and Services: CODESPECT does not warrant, endorse, or assume responsibility for any third-party products or services mentioned in this report, including any open-source or third-party software, code, libraries, materials, or information that may be linked to, referenced by, or accessible through this report. CODESPECT is not responsible for monitoring any transactions between you and third-party providers. We strongly recommend conducting thorough due diligence and exercising caution when engaging with third-party products or services, just as you would for any other product or service transaction.

Further Recommendations: We advise clients to schedule a re-audit after any significant changes to the codebase to ensure ongoing security and reduce the risk of newly introduced vulnerabilities. Additionally, we recommend implementing a bug bounty program to incentivize external developers and security researchers to identify and disclose potential vulnerabilities safely and responsibly.

Disclaimer of Advice: FOR AVOIDANCE OF DOUBT, THIS REPORT, ITS CONTENT, AND ANY ASSOCIATED SERVICES OR MATERIALS SHOULD NOT BE CONSIDERED OR RELIED UPON AS FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER PROFESSIONAL ADVICE.

3 Risk Classification

Severity Level	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Table 1: Risk Classification Matrix based on Likelihood and Impact

3.1 Impact

- **High** - Results in a substantial loss of assets (more than 10%) within the protocol or causes significant disruption to the majority of users.
- **Medium** - Losses affect less than 10% globally or impact only a portion of users, but are still considered unacceptable.
- **Low** - Losses may be inconvenient but are manageable, typically involving issues like griefing attacks that can be easily resolved or minor inefficiencies such as gas costs.

3.2 Likelihood

- **High** - Very likely to occur, either easy to exploit or difficult but highly incentivized.
- **Medium** - Likely only under certain conditions or moderately incentivized.
- **Low** - Unlikely unless specific conditions are met, or there is little-to-no incentive for exploitation.

3.3 Action Required for Severity Levels

- **Critical** - Must be addressed immediately if already deployed.
- **High** - Must be resolved before deployment (or urgently if already deployed).
- **Medium** - It is recommended to fix.
- **Low** - Can be fixed if desired but is not crucial.

In addition to High, Medium, and Low severity levels, CODESPECT utilizes two other categories for findings: **Informational** and **Best Practices**.

a) **Informational** findings do not pose a direct security risk but provide useful information the audit team wants to communicate formally.

b) **Best Practices** findings indicate that certain portions of the code deviate from established smart contract development standards.

4 Executive Summary

This document presents the security assessment conducted by CODESPECT for the smart contracts of Hyperwave. Hyperwave is a liquid token that accrues yield from the Hyperliquidity Provider Vault (HLP). hwHLP lets users profit from market making and liquidations on Hypercore, while retaining their liquid assets for DeFi on HyperEVM and beyond.

This audit reviews an upgrade enabling the Hyperwave system to utilize HyperLiquid precompiles. The updated contracts introduce manager components responsible for handling accounts that interact directly with the HyperCore.

The audit was performed using:

a) Manual analysis of the codebase.

b) Dynamic analysis of smart contracts, execution testing.

CODESPECT found six points of attention, three classified as `Low` and three classified as `Informational`. All of the issues are summarised in Table 2.

Organization of the document is as follows:

- **Section 5** summarizes the audit.
- **Section 6** describes the functionality of the code in scope.
- **Section 7** presents the issues.
- **Section 8** presents the issues discovered during the fix review phase.
- **Section 9** mentions centralisation risks.
- **Section 10** discusses the documentation provided by the client for this audit.
- **Section 11** presents the compilation and tests.

Issues found:

Severity	Unresolved	Fixed	Acknowledged
Low	0	3	0
Informational	0	1	2
Total	**0**	**4**	**2**

Table 2: Summary of Unresolved, Fixed, and Acknowledged Issues

5 Audit Summary

Audit Type	Security Review
Project Name	Hyperwave
Type of Project	Liquid token protocol
Duration of Engagement	5 Days
Duration of Fix Review Phase	2 Days
Draft Report	August 18, 2025
Final Report	August 25, 2025
Repository (boring vault)	hlp-corewriter
Commit (Audit)	5a19bb4373eaf3eb57d872f81d20177fb5cf9b2b
Commit (Final)	4108aed13b26ad0cb70fc42d2dba99eec145af7c
Documentation Assessment	High
Test Suite Assessment	High
Auditors	Talfao, JecikPo

Table 3: Summary of the Audit

5.1 Scope - Audited Files

	Contract	**LoC**
1	TradeStakeManager.sol	645
2	VaultManager.sol	192
3	Configurator.sol	184
4	util/L1ReadLib.sol	151
5	util/CoreWriterEncodingLib.sol	121
6	interfaces/ITradeStakeManager.sol	96
7	util/CoreWriterDomain.sol	79
8	HyperCoreAccount.sol	75
9	interfaces/IVaultManager.sol	29
10	interfaces/IConfigurator.sol	28
11	interfaces/IHyperCoreAccount.sol	24
12	interfaces/ISupportedTokens.sol	9
13	util/RolesLib.sol	6
14	interfaces/IWHYPE.sol	6
15	interfaces/ISupportedPerps.sol	4
	Total	**1649**

5.2 Findings Overview

	Finding	Severity	Update
1	Blocked whitelist removals for de-listed accounts	Low	Fixed
2	Joint withdrawal and spot transfer whitelist could lead to locked funds	Low	Fixed
3	Receive of Native token is enabled while withdrawal is not possible	Low	Fixed
4	Introduce validation of the order size	Info	Acknowledged
5	Replace storage writes with memory ones	Info	Acknowledged

5.3 Fix Review Phase - Findings Overview

	Finding	Severity	Update
1	Possible overflow during price tolerance control	Info	Fixed

6 System Overview

The Hyperwave team introduces a new upgrade to the codebase which allows them to directly interact with HyperCore through HyperEVM contracts. All of the contracts work together to present a centralised system for managing HyperCore accounts which have their representation as contracts on HyperEVM managed by manager contracts. The whole system is maintained by two important roles:

- `GOVERNOR` - main admin role, which is responsible for all settings of the system including token configuration, whitelist management, and perp/spot market enablement.
- `OPERATOR` - role which is responsible for executing transfers, trades, bridging operations, and direct interactions with HyperCore accounts.

The system architecture implements a two-tier whitelisting mechanism where accounts must first be whitelisted at the system level, and then specific operations (spot trading, perp trading, vault operations, transfers) require additional per-account whitelisting.

6.1 Configurator

The initial part of the system is the `Configurator` contract which is responsible for managing and keeping track of enabled tokens which can be used by the managers together with spots and perps. During the enabling of tokens, the bridge system address is saved and is calculated based on the token indexes within the HyperCore system using the formula:

```
bridgeAddress = tokenIndex << 96 | 0x2000000000000000000000000000000000000000
```

The `Configurator` provides the following key functionalities:

- **Token Management**: Enables/disables ERC20 tokens and native HYPE token for use within the system.
- **Spot Association**: Associates enabled tokens with specific spot market IDs (format: spotId >= 10000).
- **Perp Management**: Enables/disables perpetual markets based on perpIndex from the HyperCore system.

The contract uses L1Read precompiles (addresses 0x0800-0x080D) to verify that tokens and markets exist on the Hyper-Liquid platform before enabling them, ensuring system integrity.

6.2 HyperCoreAccount

`HyperCoreAccount` contracts serve as individual account representations on HyperEVM that interface directly with the CoreWriter system. Each account is deployed using a beacon proxy pattern allowing for upgradeable functionality while maintaining individual ownership.

Key characteristics:

- **Owner-Only Access**: All operations require owner authorization. The owner is `TradeStakeManager` or `VaultManager` contract.
- **CoreWriter Integration**: Encodes and sends protocol actions to HyperCore system via CoreWriter.
- **Asset Management**: Handles both ERC20 token transfers and native HYPE transfer.
- **Trading Operations**: Supports limit orders, vault transfers, staking, delegation, and spot transfers.

The account contracts act as the execution layer, translating manager commands into CoreWriter protocol actions.

6.3 Managers

The system contains two managers `TradeStakeManager` and `VaultManager`, which are responsible for managing the `HyperCoreAccounts` through a comprehensive access control and whitelisting system.

6.3.1 TradeStakeManager

`TradeStakeManager` is the primary manager through which asset flows occur. The Hyperwave managed bot calls `forward(...)` or
`forwardNative(...)` functions through the boring vault, which then transfers assets from this vault to `HyperCoreAccount` and bridges them to the HyperCore system.

Core functionalities include:

- **Spot Trading**: Buy/sell orders with configurable price bounds and token-specific whitelisting
- **Perp Trading**: Perpetual trading with per-account perp whitelisting and reduce-only options

- **Staking Operations**: Staking deposits and withdrawals for yield generation.
- **Delegation Management**: Token delegation to validators with validator-specific whitelisting.
- **Cross-class Transfers**: Moving funds between spot and perp trading accounts.
- **Asset Bridging**: Forward/bridge operations from L1 to HyperCore system.
- **EVM Withdrawals**: Withdraw assets from HyperCore back to EVM addresses.

The contract enforces price bounds for spot trading with separate minimum/maximum limits for buy and sell orders, providing risk management controls.

6.3.2 VaultManager

`VaultManager` focuses specifically on vault operations and provides a subset of functionality compared to `TradeStakeManager`:

- **Vault Operations**: Deposits and withdrawals from yield-generating vaults
- **Cross-class Transfers**: USD transfers between spot and perp accounts
- **Spot Transfers**: Token transfers with destination whitelisting
- **Simplified Access Control**: Two-tier whitelisting (account + vault/destination specific)

The `VaultManager` is designed for vault-specific operations and does not handle trading, staking, or bridging functionalities, providing a more focused and simplified interface.

6.4 System Flow

The typical asset flow follows this pattern:

a. External assets (from Boring Vault) are forwarded to `HyperCoreAccount` via `TradeStakeManager`
b. Assets are bridged from HyperEVM to HyperCore system
c. Trading, staking, or vault operations are executed on HyperCore
d. Cross-account transfers can move assets between different `HyperCoreAccounts`
e. Assets can be withdrawn back to EVM addresses when needed

Both managers implement comprehensive safety mechanisms including reentrancy protection, role-based access control, multi-tier whitelisting, and configuration validation to ensure secure operations across the entire system.

7 Issues

7.1 [Low] Blocked whitelist removals for de-listed accounts

File(s): `TradeStakeManager.sol` `VaultManager.sol`

Description: Both `TradeStakeManager` and `VaultManager` maintain separate whitelists to control different operations:

1. `TradeStakeManager` maintains `SpotWhitelist`, `PerpWhitelist`, `ValidatorWhitelist` and `TransferWhitelist`;
2. `VaultManager` maintains `TransferWhitelist` and `VaultWhitelist`;

Those whitelists allow whitelisting of different indexes or addresses per each `HyperCoreAccount` address. It is not possible to update any of those lists without first having the the concrete `HyperCoreAccount` address whitelisted through a separate whitelist - `AccountWhitelist` first. This is also valid for de-listing. Once a `HyperCoreAccount` is de-listed it is not possible to delist anything associated with that account from the above whitelists.

The problem might arise if a `HyperCoreAccount` is de-listed (e.g. due to emergency reasons) and needs to be listed back, yet without certain perps, or indexes, it cannot be done. It must first be whitelisted and only then the perps or indexes can be delisted.

Impact: Inability to whitelist an account without previously delisting problematic assets.

Recommendation(s): Allow whitelist removals without checking first if the `HyperCoreAccount` is whitelisted.

Status: Fixed

Update from Hyperwave: Resolved in 7a406f9e70161ad69884eac8e2f6f4ae84b2cbe1

7.2 [Low] Joint withdrawal and spot transfer whitelist could lead to locked funds

File(s): `TradeStakeManager.sol`

Description: The `TradeStakeManager` contract uses `transferWhitelist` to validate both:

1. withdrawals from the account's owned L1 address using `withdraw(...)` and `withdrawNative(...)`;
2. spot transfers using `spotSend(...)` and `spotSendUSDC(...)`;

This leads to a situation where a valid destination account used for `withdraw(...)` or `withdrawNative(...)` could be passed to the `spotSend(...)` or `spotSendUSDC(...)` and still be allowed. This could lead to transfer of the spot balance to an account which the protocol doesn't have control of, and hence lead to locked funds. While this mistake would have to be made through an off-chain mechanism or the `BoringVault`, which is outside of the scope, it is worth raising the issue due to tight design control of the flow transfer within the protocol.

Impact: Potential loss of transferred funds.

Recommendation(s): Create separate whitelists for spot transfers and withdrawals

Status: Fixed

Update from Hyperwave: Resolved in 4b75b10a6aa5e1b59ed000bc7379af10eacefd67

7.3 [Low] Receive of Native token is enabled while withdrawal is not possible

File(s): `TradeStakeManager.sol`

Description: The `TradeStakeManager` contract contains a `receive()` function which allows receiving of the Native HYPE token by the contract.

There is however no option to withdraw the excess Native asset from the `TradeStakeManager`. The existing `withdrawNative(...)` function allows withdrawal of HYPE from `HyperCoreAccount` only.

Impact: Native HYPE asset transferred to the contract cannot be withdrawn.

Recommendation(s): Add a function allowing withdrawal of excess HYPE balance from the `TradeStakeManager` contract.

Status: Fixed

Update from Hyperwave: Resolved in 0096241fb91c1158c3b3526a06049bad08c7b0eb

7.4 [Info] Introduce validation of the order size

File(s): TradeStakeManager.sol

Description: The Hyperliquid core enforces decimal limits on order sizes for both perps and spots.

According to the documentation: *Prices can have up to 5 significant figures, but no more than MAX_DECIMALS - szDecimals decimal places, where MAX_DECIMALS is 6 for perps and 8 for spot.* [ref]

Implementing similar validation in TradeStakeManager would help prevent potential silent failures when an incorrect order size is submitted. While a similar limitation already exists for prices — and TradeStakeManager enforces price boundaries — the same should be applied to order sizes.

Impact: Proper validation can prevent unnecessary silent failures on the Hyperliquid Core side.

Recommendation(s): Validate order sizes in accordance with Hyperliquid documentation to ensure compliance before submission.

Status: Acknowledged

Update from Hyperwave: We acknowledge and choose not to implement in the contract. If invalid, hypercore will reject the request. There are many other validations we could do and this fits into the same category

7.5 [Info] Replace storage writes with memory ones

File(s): TradeStakeManager.sol

Description: Whenever the bounds for native or other tokens are queried, they are assigned to a variable declared as storage, e.g.:

```
PriceBounds storage bounds; // @audit Is storage necessary here?
if (requestedNative) {
    spotWhitelisted = _getNativeSpotWhitelist(account);
    bounds = _getNativePriceBounds();
} else {
    spotWhitelisted = _getTokenSpotWhitelist(account, tokenAddress);
    bounds = _getTokenPriceBounds(tokenAddress);
}
```

The _getTokenPriceBounds(...) function returns a storage variable. However, in functions such as placeSellSpotOrder(...) or placeBuySpotOrder(...), the bounds are only read, not modified. Using storage in these cases is unnecessary and increases gas usage. A getter returning a memory copy of the bounds would be more efficient.

Impact: Minor gas optimisation; limited effect due to the use of Hyperliquid.

Recommendation(s): Introduce getter functions that return the bounds as memory when only reading data, avoiding unnecessary storage references.

Status: Acknowledged

Update from Hyperwave: We acknowledge and choose not to change, since the benefits are eroded by copying unused fields from storage.

8 Fix Review Phase Issues

8.1 [Info] Possible overflow during price tolerance control

File(s): TradeStakeManager.sol

Description: The 6dabdeeec70afc123d694fac0fcbb2d90ddb4c87 commit introduced control on perps price using tolerance parameter which can be set by the GOVERNOR role for each perpIndex. The price privided by the OPERATOR during sell and buy order placing is compared against markPrice which is obtained from the HyperCore and is scaled to 8 decimals by the _getMarkPxForPerp(...) function.

Below minPrice and maxPrice calculation can revert during the multiplication part, especially for the last one. In case the markPrice * (10000 + toleranceBps) crosses the type(uint64).max value, overflow happens.

```
uint64 markPrice = _getMarkPxForPerp(perpIndex);
uint64 minPrice = (markPrice * (10000 - toleranceBps)) / 10000;
uint64 maxPrice = (markPrice * (10000 + toleranceBps)) / 10000;
```

As the markPrice is checked inside _getMarkPxForPerp(...) to be lower than type(uint64).max, such situation can theoretically happen, If the returned markPrice is above 922383322851620. That is assuming the maximum possible toleranceBps value, which is 19_999.

Impact: Order placement will revert in case the price is sufficiently high. The above number, when interpreted as an 8 decimal USDC price, is $9.223.372 hence it is not likely to happen in the near future; however, if the HyperCore introduces other quote currencies in the future, it might become a problem.

Recommendation(s): Introduce uint256 casting to the operation, before casting it down to uint64.

Status: Fixed

Update from HyperWave: Resolved in 4108aed13b26ad0cb70fc42d2dba99eec145af7c

9 Centralisation Risks

The project is highly centralised, with overall control of all assets vested in the `GOVERNOR` role. It is therefore essential for the protocol team to implement a robust key management system and a secure multi-signature setup to minimise the risk of system compromise.

The `OPERATOR` role is also critical. As it will likely be a “boring vault” managed by an automated bot (back-end system), strong key management practices and thorough server hardening are required. We also strongly recommend implementing comprehensive monitoring of all operator actions to promptly detect and mitigate any malicious trades that could endanger user funds.

10 Evaluation of Provided Documentation

The Hyperwave documentation was primarily delivered through **NatSpec** comments within the Solidity contracts. These comments were helpful in understanding the overall functionality of the protocol and were particularly effective in areas where they explained specific design decisions.

The protocol team supplied additional information in the form of Notion articles and components interaction diagrams. The Notion articles described HyperCore position management, which is executed by the Vault and Trade Manager contracts. Those are practical in terms of documententing the assumptions and limitations on managing the HyperCore orders and funds transfer between HyperCore and HyperEVM. While the graphs included a visual representation of interactions between different system components.

The Hyperwave team remained consistently available and responsive, promptly addressing all questions and concerns raised by **CODESPECT** during the audit process.

11 Test Suite Evaluation

11.1 Compilation Output

```
> forge build
[] Compiling...
[] Compiling 153 files with Solc 0.8.29
[] Solc 0.8.29 finished in 3.27s
Compiler run successful with warnings:
[...]
```

11.2 Tests Output

```
% forge test
[] Compiling...
No files changed, compilation skipped

Ran 1 test for test/integration/scenarios/StakingValidatorMigration.t.sol:StakingValidatorMigrationTest
[PASS] test_stakingValidatorMigration() (gas: 307437)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 11.31ms (795.83µs CPU time)

Ran 4 tests for test/unit/class-transfers/TradeStakeManagerPerpToSpot.t.sol:TradeStakeManagerPerpToSpotTest
[PASS] test_perpToSpot_callsUsdClassTransfer() (gas: 221782)
[PASS] test_perpToSpot_multipleTransfers() (gas: 521113)
[PASS] test_perpToSpot_nonWhitelistedAccount_reverts() (gas: 3685720)
[PASS] test_perpToSpot_zeroAmount_reverts() (gas: 25416)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 11.40ms (2.60ms CPU time)

Ran 1 test for test/integration/scenarios/BeaconHandover.t.sol:BeaconHandoverTest
[PASS] test_completeBeaconHandoverFlow() (gas: 10790840)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 11.58ms (3.44ms CPU time)

Ran 1 test for test/integration/scenarios/TokenDeprecation.t.sol:TokenDeprecationTest
[PASS] test_tokenDeprecationFlow() (gas: 513146)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 11.65ms (568.08µs CPU time)

Ran 9 tests for test/unit/perps-support/PerpRegistration.t.sol:PerpRegistrationTest
[PASS] test_disablePerp_notEnabled_reverts() (gas: 25642)
[PASS] test_disablePerp_success() (gas: 47016)
[PASS] test_enablePerp_afterSettingUpPerp_succeeds() (gas: 96207)
[PASS] test_enablePerp_alreadyEnabled_reverts() (gas: 56670)
[PASS] test_enablePerp_nonExistentPerp_reverts() (gas: 28601)
[PASS] test_enablePerp_success() (gas: 58620)
[PASS] test_isPerpSupported_queries() (gas: 116709)
[PASS] test_perpLifecycle_enableDisableReenable() (gas: 80504)
[PASS] test_zeroIndex_enableDisablePerp() (gas: 81592)
Suite result: ok. 9 passed; 0 failed; 0 skipped; finished in 11.64ms (2.50ms CPU time)

Ran 2 tests for test/integration/scenarios/ManagerReplacement.t.sol:ManagerReplacementTest
[PASS] test_tradeStakeManagerReplacement() (gas: 1162912)
[PASS] test_vaultManagerReplacement() (gas: 962041)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 11.70ms (1.24ms CPU time)

Ran 6 tests for test/unit/vaults/VaultDeposits.t.sol:VaultDepositsTest
[PASS] test_vaultDeposit_multipleDeposits_success() (gas: 470530)
[PASS] test_vaultDeposit_multipleVaults_success() (gas: 504979)
[PASS] test_vaultDeposit_nonWhitelistedAccount_reverts() (gas: 27509)
[PASS] test_vaultDeposit_nonWhitelistedVault_reverts() (gas: 30842)
[PASS] test_vaultDeposit_success() (gas: 276943)
[PASS] test_vaultDeposit_zeroAmount_reverts() (gas: 27847)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 12.15ms (3.07ms CPU time)


Ran 15 tests for test/unit/tokens-support/TokenRegistration.t.sol:TokenRegistrationTest
[PASS] test_associateSpot_undefinedToken_reverts() (gas: 25241)
[PASS] test_bridgeAddressCalculation_multipleTokens() (gas: 288114)
[PASS] test_bridgeAddressCalculation_withoutL1BridgeLib() (gas: 88283)
[PASS] test_disableToken_native() (gas: 75773)
[PASS] test_disableToken_notEnabled_reverts() (gas: 22653)
[PASS] test_enableToken_alreadyEnabled_reverts() (gas: 91902)
[PASS] test_enableToken_zeroAddressFromPrecompile_reverts() (gas: 33398)
[PASS] test_getBridgeAddress_erc20_calculatesAddress() (gas: 124721)
[PASS] test_getBridgeAddress_native_returnsConstant() (gas: 122508)
[PASS] test_getToken_notEnabled_returnsEmpty() (gas: 22276)
[PASS] test_registerNative_success() (gas: 123780)
[PASS] test_registerToken_allocatedIndex_success() (gas: 125980)
[PASS] test_registerToken_differentAddress_success() (gas: 126003)
[PASS] test_tokenLifecycle_enableDisableClear() (gas: 150900)
[PASS] test_updateSpotId_success() (gas: 165378)
Suite result: ok. 15 passed; 0 failed; 0 skipped; finished in 12.67ms (4.01ms CPU time)

Ran 9 tests for test/unit/staking/StakingWithdrawals.t.sol:StakingWithdrawalsTest
[PASS] test_withdrawStaking_amountVariations() (gas: 803105)
[PASS] test_withdrawStaking_depositWithdrawSequence() (gas: 673976)
[PASS] test_withdrawStaking_largeAmount_success() (gas: 214953)
[PASS] test_withdrawStaking_maxValue() (gas: 214867)
```

```
[PASS] test_withdrawStaking_multipleAmounts() (gas: 644680)
[PASS] test_withdrawStaking_multipleWithdrawals() (gas: 508005)
[PASS] test_withdrawStaking_notWhitelistedAccount_reverts() (gas: 3687653)
[PASS] test_withdrawStaking_validParams_success() (gas: 214909)
[PASS] test_withdrawStaking_zeroAmount_reverts() (gas: 25213)
Suite result: ok. 9 passed; 0 failed; 0 skipped; finished in 12.68ms (3.92ms CPU time)

Ran 4 tests for test/unit/class-transfers/TradeStakeManagerSpotToPerp.t.sol:TradeStakeManagerSpotToPerpTest
[PASS] test_spotToPerp_callsUsdClassTransfer() (gas: 241616)
[PASS] test_spotToPerp_largeAmount() (gas: 241649)
[PASS] test_spotToPerp_nonWhitelistedAccount_reverts() (gas: 3685738)
[PASS] test_spotToPerp_zeroAmount_reverts() (gas: 25392)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 1.02ms (201.04µs CPU time)

Ran 3 tests for
 → test/unit/spot-transfers/TradeStakeManagerTokenSpotTransfer.t.sol:TradeStakeManagerTokenSpotTransferTest
[PASS] test_spotSend_multipleTokenTypes() (gas: 759344)
[PASS] test_spotSend_nonUsdcToken_success() (gas: 309786)
[PASS] test_spotSend_unregisteredToken_reverts() (gas: 40332)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 1.14ms (278.96µs CPU time)

Ran 6 tests for test/unit/whitelists/TradeStakeManagerAccountWhitelist.t.sol:TradeStakeManagerAccountWhitelistTest
[PASS] test_accountWhitelist_requiredForOtherWhitelists() (gas: 275323)
[PASS] test_setAccountWhitelist_add_success() (gas: 54877)
[PASS] test_setAccountWhitelist_duplicate_idempotent() (gas: 57861)
[PASS] test_setAccountWhitelist_independentAccounts() (gas: 7403891)
[PASS] test_setAccountWhitelist_remove_success() (gas: 44369)
[PASS] test_setAccountWhitelist_zeroAddress_reverts() (gas: 20430)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 1.34ms (610.71µs CPU time)

Ran 1 test for test/integration/scenarios/TradeStakeAccountMigration.t.sol:TradeStakeAccountMigrationTest
[PASS] test_tradeStakeAccountMigration() (gas: 928938)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 14.11ms (3.81ms CPU time)

Ran 8 tests for test/unit/whitelists/VaultManagerTransferWhitelist.t.sol:VaultManagerTransferWhitelistTest
[PASS] test_setTransferWhitelist_add_success() (gas: 58071)
[PASS] test_setTransferWhitelist_duplicate_idempotent() (gas: 66205)
[PASS] test_setTransferWhitelist_multipleDestinations() (gas: 387498)
[PASS] test_setTransferWhitelist_remove_success() (gas: 55659)
[PASS] test_setTransferWhitelist_revertsForNonWhitelistedAccount() (gas: 27568)
[PASS] test_setTransferWhitelist_zeroAccount_reverts() (gas: 25411)
[PASS] test_setTransferWhitelist_zeroDestination_reverts() (gas: 25635)
[PASS] test_transferWhitelist_accountIsolation() (gas: 294844)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 2.59ms (1.72ms CPU time)

Ran 24 tests for test/unit/access/HyperCoreAccountAccess.t.sol:HyperCoreAccountAccessTest
[PASS] test_cancelOrderByOid_notOwner_reverts() (gas: 23325)
[PASS] test_cancelOrderByOid_onlyOwner_success() (gas: 54829)
[PASS] test_initialize_beaconImplementation_reverts() (gas: 11409)
[PASS] test_limitOrder_notOwner_reverts() (gas: 24097)
[PASS] test_limitOrder_onlyOwner_success() (gas: 61537)
[PASS] test_spotSend_notOwner_reverts() (gas: 25643)
[PASS] test_spotSend_onlyOwner_success() (gas: 58543)
[PASS] test_stakingDeposit_notOwner_reverts() (gas: 23209)
[PASS] test_stakingDeposit_onlyOwner_success() (gas: 53531)
[PASS] test_stakingWithdraw_notOwner_reverts() (gas: 23166)
[PASS] test_stakingWithdraw_onlyOwner_success() (gas: 53556)
[PASS] test_tokenDelegate_notOwner_reverts() (gas: 25665)
[PASS] test_tokenDelegate_onlyOwner_success() (gas: 58543)
[PASS] test_transferERC20_notOwner_reverts() (gas: 63478)
[PASS] test_transferERC20_onlyOwner_success() (gas: 66918)
[PASS] test_transferNative_notOwner_reverts() (gas: 42628)
[PASS] test_transferNative_onlyOwner_success() (gas: 61463)
[PASS] test_upgrade_affectsAllProxies() (gas: 2615856)
[PASS] test_upgrade_notAccountOwner_reverts() (gas: 2072621)
[PASS] test_upgrade_onlyBeaconOwner_success() (gas: 2113057)
[PASS] test_usdClassTransfer_notOwner_reverts() (gas: 23346)
[PASS] test_usdClassTransfer_onlyOwner_success() (gas: 54883)
[PASS] test_vaultTransfer_notOwner_reverts() (gas: 25622)
[PASS] test_vaultTransfer_onlyOwner_success() (gas: 58521)
Suite result: ok. 24 passed; 0 failed; 0 skipped; finished in 14.42ms (5.56ms CPU time)

Ran 6 tests for test/unit/spot-transfers/TradeStakeManagerUsdcSpotTransfer.t.sol:TradeStakeManagerUsdcSpotTransferTest
[PASS] test_spotSendUSDC_notWhitelistedDestination_reverts() (gas: 30886)
```

```
[PASS] test_spotSendUSDC_success() (gas: 275543)
[PASS] test_spotSend_nonWhitelistedAccount_reverts() (gas: 3690681)
[PASS] test_spotSend_nonWhitelistedDestination_reverts() (gas: 43620)
[PASS] test_spotSend_success() (gas: 275523)
[PASS] test_spotSend_zeroAmount_reverts() (gas: 30354)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 1.20ms (331.75µs CPU time)

Ran 5 tests for test/unit/spot-transfers/VaultManagerUsdcSpotTransfer.t.sol:VaultManagerUsdcSpotTransferTest
[PASS] test_spotSendUSDC_success() (gas: 275454)
[PASS] test_spotSend_nonWhitelistedAccount_reverts() (gas: 3690614)
[PASS] test_spotSend_nonWhitelistedDestination_reverts() (gas: 43553)
[PASS] test_spotSend_success() (gas: 275434)
[PASS] test_spotSend_zeroAmount_reverts() (gas: 30287)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 1.04ms (284.63µs CPU time)

Ran 8 tests for test/unit/whitelists/TradeStakeManagerTransferWhitelist.t.sol:TradeStakeManagerTransferWhitelistTest
[PASS] test_setTransferWhitelist_add_success() (gas: 58028)
[PASS] test_setTransferWhitelist_duplicate_idempotent() (gas: 66119)
[PASS] test_setTransferWhitelist_multipleDestinations() (gas: 387068)
[PASS] test_setTransferWhitelist_remove_success() (gas: 55661)
[PASS] test_setTransferWhitelist_revertsForNonWhitelistedAccount() (gas: 27590)
[PASS] test_setTransferWhitelist_zeroAccount_reverts() (gas: 25433)
[PASS] test_setTransferWhitelist_zeroDestination_reverts() (gas: 25657)
[PASS] test_transferWhitelist_accountIsolation() (gas: 294825)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 1.72ms (1.01ms CPU time)

Ran 8 tests for test/unit/whitelists/VaultManagerAccountWhitelist.t.sol:VaultManagerAccountWhitelistTest
[PASS] test_accountWhitelist_managerIndependence() (gas: 7101160)
[PASS] test_removeAccountsFromWhitelist_batch_success() (gas: 166943)
[PASS] test_setAccountWhitelist_add_success() (gas: 54743)
[PASS] test_setAccountWhitelist_duplicate_idempotent() (gas: 57617)
[PASS] test_setAccountWhitelist_multipleAccounts() (gas: 358644)
[PASS] test_setAccountWhitelist_remove_success() (gas: 44174)
[PASS] test_setAccountWhitelist_zeroAddress_reverts() (gas: 20341)
[PASS] test_whitelistAccounts_batch_success() (gas: 1282459)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 2.27ms (1.57ms CPU time)

Ran 4 tests for test/unit/class-transfers/VaultManagerPerpToSpot.t.sol:VaultManagerPerpToSpotTest
[PASS] test_perpToSpot_callsUsdClassTransfer() (gas: 221715)
[PASS] test_perpToSpot_multipleTransfers() (gas: 520912)
[PASS] test_perpToSpot_nonWhitelistedAccount_reverts() (gas: 3685653)
[PASS] test_perpToSpot_zeroAmount_reverts() (gas: 25349)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 790.08µs (255.04µs CPU time)

Ran 4 tests for test/unit/class-transfers/VaultManagerSpotToPerp.t.sol:VaultManagerSpotToPerpTest
[PASS] test_spotToPerp_callsUsdClassTransfer() (gas: 241572)
[PASS] test_spotToPerp_largeAmount() (gas: 241605)
[PASS] test_spotToPerp_nonWhitelistedAccount_reverts() (gas: 3685694)
[PASS] test_spotToPerp_zeroAmount_reverts() (gas: 25348)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 719.63µs (195.50µs CPU time)

Ran 3 tests for test/unit/spot-transfers/VaultManagerTokenSpotTransfer.t.sol:VaultManagerTokenSpotTransferTest
[PASS] test_spotSend_multipleTokenTypes() (gas: 759143)
[PASS] test_spotSend_nonUsdcToken_success() (gas: 309719)
[PASS] test_spotSend_unregisteredToken_reverts() (gas: 40265)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 1.03ms (282.17µs CPU time)

Ran 1 test for test/integration/flows/PerpsFlow.t.sol:PerpsFlowTest
[PASS] test_perpOrderFlow() (gas: 987788)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 3.95ms (1.60ms CPU time)

Ran 1 test for test/integration/flows/VaultFlow.t.sol:VaultFlowTest
[PASS] test_completeVaultFlow() (gas: 774739)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 8.06ms (1.22ms CPU time)

Ran 3 tests for test/integration/scenarios/ProxyAdminTransfer.t.sol:ProxyAdminTransferTest
[PASS] test_proxyAdminTransfer_Configurator() (gas: 8192746)
[PASS] test_proxyAdminTransfer_TradeStakeManager() (gas: 22399932)
[PASS] test_proxyAdminTransfer_VaultManager() (gas: 11660265)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 1.98ms (1.40ms CPU time)

Ran 1 test for test/integration/scenarios/VaultMigration.t.sol:VaultMigrationTest
[PASS] test_completeVaultMigration() (gas: 580784)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 11.78ms (3.38ms CPU time)
```

```
Ran 9 tests for test/unit/whitelists/PerpWhitelist.t.sol:PerpWhitelistTest
[PASS] test_cancelPerpOrder_requiresWhitelist() (gas: 340294)
[PASS] test_getPerpWhitelist() (gas: 110131)
[PASS] test_perpWhitelist_independentFromSpotWhitelist() (gas: 405696)
[PASS] test_perpWhitelist_withWhitelistSetupLib() (gas: 667664)
[PASS] test_setPerpWhitelist_disables_trading() (gas: 360691)
[PASS] test_setPerpWhitelist_enables_trading() (gas: 374838)
[PASS] test_setPerpWhitelist_multiple_perps() (gas: 943103)
[PASS] test_setPerpWhitelist_nonEnabledPerp_reverts() (gas: 33551)
[PASS] test_setPerpWhitelist_revertsForNonWhitelistedAccount() (gas: 25329)
Suite result: ok. 9 passed; 0 failed; 0 skipped; finished in 3.27ms (1.84ms CPU time)

Ran 8 tests for test/unit/staking/Undelegation.t.sol:UndelegationTest
[PASS] testFuzz_undelegate_variousAmounts(uint64) (runs: 256, : 271039, ~: 271039)
[PASS] test_undelegate_maxValues() (gas: 300529)
[PASS] test_undelegate_multipleAmounts_sameValidator() (gas: 672176)
[PASS] test_undelegate_multipleValidators() (gas: 760010)
[PASS] test_undelegate_notWhitelistedAccount_reverts() (gas: 3690199)
[PASS] test_undelegate_notWhitelistedValidator_reverts() (gas: 28760)
[PASS] test_undelegate_whitelistedValidator_success() (gas: 271523)
[PASS] test_undelegate_zeroAmount_reverts() (gas: 27799)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 13.79ms (13.12ms CPU time)

Ran 2 tests for test/integration/scenarios/RoleAddition.t.sol:RoleAdditionTest
[PASS] test_roleAddition_TradeStakeManager() (gas: 6136544)
[PASS] test_roleAddition_VaultManager() (gas: 2577695)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 7.47ms (1.29ms CPU time)

Ran 24 tests for test/unit/access/VaultManagerAccess.t.sol:VaultManagerAccessTest
[PASS] test_initialization_grantsOperatorRoleToGovernor() (gas: 3458447)
[PASS] test_initialization_rolesSetCorrectly() (gas: 3500962)
[PASS] test_initialize_implementation_reverts() (gas: 16141)
[PASS] test_initialize_onlyCallableOnce() (gas: 47010)
[PASS] test_multipleAccounts_independence() (gas: 701139)
[PASS] test_perpToSpot_onlyOperator() (gas: 143149)
[PASS] test_publicGetters_anyoneCanCall() (gas: 63326)
[PASS] test_roleManagement_governorCanManageOperators() (gas: 97333)
[PASS] test_roleTransitions() (gas: 131772)
[PASS] test_setAccountWhitelist_onlyGovernor() (gas: 52403)
[PASS] test_setTransferWhitelist_false_onlyGovernor() (gas: 57668)
[PASS] test_setTransferWhitelist_onlyGovernor() (gas: 78961)
[PASS] test_setVaultWhitelist_false_onlyGovernor() (gas: 57580)
[PASS] test_setVaultWhitelist_onlyGovernor() (gas: 78939)
[PASS] test_spotSendUSDC_onlyOperator() (gas: 174513)
[PASS] test_spotSendUSDC_requiresWhitelist() (gas: 141061)
[PASS] test_spotSend_onlyOperator() (gas: 220343)
[PASS] test_spotToPerp_onlyOperator() (gas: 143214)
[PASS] test_upgrade_notProxyAdminOwner_reverts() (gas: 2437717)
[PASS] test_upgrade_onlyProxyAdmin_success() (gas: 2461687)
[PASS] test_vaultDeposit_onlyOperator() (gas: 174358)
[PASS] test_vaultDeposit_requiresWhitelist() (gas: 140973)
[PASS] test_vaultWithdraw_onlyOperator() (gas: 174502)
[PASS] test_vaultWithdraw_requiresWhitelist() (gas: 141099)
Suite result: ok. 24 passed; 0 failed; 0 skipped; finished in 10.55ms (9.10ms CPU time)

Ran 10 tests for test/unit/whitelists/ValidatorWhitelist.t.sol:ValidatorWhitelistTest
[PASS] test_setValidatorWhitelist_add_success() (gas: 61917)
[PASS] test_setValidatorWhitelist_duplicate_idempotent() (gas: 66076)
[PASS] test_setValidatorWhitelist_independentLists() (gas: 7589339)
[PASS] test_setValidatorWhitelist_multipleValidators() (gas: 203799)
[PASS] test_setValidatorWhitelist_remove_success() (gas: 51264)
[PASS] test_setValidatorWhitelist_revertsForNonWhitelistedAccount() (gas: 27569)
[PASS] test_setValidatorWhitelist_withConfig() (gas: 3818809)
[PASS] test_setValidatorWhitelist_zeroValidator_reverts() (gas: 25614)
[PASS] test_validatorWhitelist_accountIsolation() (gas: 3750877)
[PASS] test_validatorWhitelist_removeAndReAdd() (gas: 156170)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 2.32ms (1.56ms CPU time)

Ran 7 tests for test/unit/bridging/BridgeFromL1.t.sol:BridgeFromL1Test
[PASS] testFuzz_bridgeFromL1_variousAmounts(uint256) (runs: 256, : 300459, ~: 300459)
[PASS] test_bridgeFromL1_multipleTokensSequentially() (gas: 528038)
[PASS] test_bridgeFromL1_nativeToken() (gas: 297599)
[PASS] test_bridgeFromL1_nonNativeToken() (gas: 299710)
[PASS] test_bridgeFromL1_revertsForNonWhitelistedAccount() (gas: 27486)
```

```
[PASS] test_bridgeFromL1_revertsForUnregisteredToken() (gas: 37866)
[PASS] test_bridgeFromL1_zeroAmount_reverts() (gas: 27802)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 32.79ms (23.95ms CPU time)

Ran 17 tests for test/unit/access/ConfiguratorAccess.t.sol:ConfiguratorAccessTest
[PASS] test_associateSpot_onlyGovernor() (gas: 137634)
[PASS] test_disablePerp_onlyGovernor() (gas: 55478)
[PASS] test_disableToken_onlyGovernor() (gas: 83206)
[PASS] test_disassociateSpot_onlyGovernor() (gas: 142860)
[PASS] test_enablePerp_onlyGovernor() (gas: 78213)
[PASS] test_enableToken_onlyGovernor() (gas: 106932)
[PASS] test_governorOnlyFunctions_exhaustive() (gas: 34018)
[PASS] test_initialization_operatorMustBeGrantedExplicitly() (gas: 3439883)
[PASS] test_initialization_rolesSetCorrectly() (gas: 3426816)
[PASS] test_initialize_implementation_reverts() (gas: 13811)
[PASS] test_initialize_onlyCallableOnce() (gas: 44662)
[PASS] test_publicGetters_anyoneCanCall() (gas: 154388)
[PASS] test_roleManagement_defaultAdminRole() (gas: 71945)
[PASS] test_roleRenouncement() (gas: 30926)
[PASS] test_roleRevocation() (gas: 110236)
[PASS] test_upgrade_notProxyAdminOwner_reverts() (gas: 2438303)
[PASS] test_upgrade_onlyProxyAdmin_success() (gas: 2462295)
Suite result: ok. 17 passed; 0 failed; 0 skipped; finished in 2.10ms (1.58ms CPU time)

Ran 1 test for test/integration/scenarios/VaultAccountMigration.t.sol:VaultAccountMigrationTest
[PASS] test_vaultAccountMigration() (gas: 809956)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 3.63ms (1.29ms CPU time)

Ran 15 tests for test/unit/tokens-support/SpotAssociation.t.sol:SpotAssociationTest
[PASS] test_associateSpot_alreadyAssociated_reverts() (gas: 124222)
[PASS] test_associateSpot_nonExistentSpot_reverts() (gas: 91830)
[PASS] test_disassociateSpot_fromNonZeroToZero() (gas: 134131)
[PASS] test_disassociateSpot_notAssociated_reverts() (gas: 84941)
[PASS] test_invalidSpotIdFormat_belowMinimum() (gas: 129490)
[PASS] test_reassociateSpot_sameValue() (gas: 141066)
[PASS] test_spotAssociation_storageIntegrity() (gas: 175524)
[PASS] test_spotId_independentOfTokenIndex() (gas: 538315)
[PASS] test_spotId_maxUint32_allowed() (gas: 124485)
[PASS] test_spotId_multipleTokens_sameSpotId() (gas: 285703)
[PASS] test_spotId_multipleTokens_uniqueSpotIds() (gas: 329969)
[PASS] test_spotId_multipleUpdates() (gas: 230627)
[PASS] test_spotId_nativeToken_specialHandling() (gas: 167863)
[PASS] test_tokenNotFoundInSpotInfo_mismatch() (gas: 119223)
[PASS] test_updateSpotId_fromNonZeroToNonZero() (gas: 170419)
Suite result: ok. 15 passed; 0 failed; 0 skipped; finished in 6.22ms (5.58ms CPU time)

Ran 2 tests for test/integration/scenarios/VaultSpotRecovery.t.sol:VaultSpotRecoveryTest
[PASS] test_vaultSpotRecovery_LST() (gas: 453826)
[PASS] test_vaultSpotRecovery_Stable() (gas: 468391)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 8.54ms (2.92ms CPU time)

Ran 14 tests for test/unit/whitelists/VaultWhitelist.t.sol:VaultWhitelistTest
[PASS] test_getVaultWhitelist_multipleVaults() (gas: 98159)
[PASS] test_setAccountWhitelist_emitsEvent() (gas: 40306)
[PASS] test_setVaultWhitelist_alreadyWhitelisted_succeeds() (gas: 62101)
[PASS] test_setVaultWhitelist_emitsEvent() (gas: 47200)
[PASS] test_setVaultWhitelist_notWhitelisted_succeeds() (gas: 35191)
[PASS] test_setVaultWhitelist_revertsForNonWhitelistedAccount() (gas: 27607)
[PASS] test_setVaultWhitelist_zeroAccount_reverts() (gas: 25411)
[PASS] test_setVaultWhitelist_zeroVault_reverts() (gas: 25568)
[PASS] test_spotSend_anyToken() (gas: 362665)
[PASS] test_spotSend_notWhitelistedAccount_reverts() (gas: 3725499)
[PASS] test_vaultDeposit_notWhitelistedAccount_reverts() (gas: 3688126)
[PASS] test_vaultDeposit_requiresWhitelist() (gas: 316905)
[PASS] test_vaultWhitelist_accountIsolation() (gas: 87982)
[PASS] test_vaultWithdraw_requiresWhitelist() (gas: 297066)
Suite result: ok. 14 passed; 0 failed; 0 skipped; finished in 1.93ms (1.29ms CPU time)

Ran 10 tests for test/unit/bridging/ForwardNative.t.sol:ForwardNativeTest
[PASS] test_bridge_deposit_emits_receive() (gas: 28258)
[PASS] test_direct_eth_transfer_emits_receive() (gas: 27373)
[PASS] test_forward_native_exactBalance() (gas: 331492)
[PASS] test_forward_native_insufficientBalance_reverts() (gas: 50331)
[PASS] test_forward_native_largeAmount() (gas: 331453)
```

```
[PASS] test_forward_native_multipleForwards() (gas: 1028488)
[PASS] test_forward_native_nonWhitelistedAccount_reverts() (gas: 3749779)
[PASS] test_forward_native_success() (gas: 334768)
[PASS] test_forward_native_tokenNotConfigured_reverts() (gas: 11167622)
[PASS] test_forward_native_zeroAmount_reverts() (gas: 30490)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 2.21ms (1.31ms CPU time)

Ran 7 tests for test/unit/vaults/VaultWithdrawals.t.sol:VaultWithdrawalsTest
[PASS] test_vaultDepositWithdraw_sequence() (gas: 841740)
[PASS] test_vaultWithdraw_multipleVaults_success() (gas: 506124)
[PASS] test_vaultWithdraw_multipleWithdrawals_success() (gas: 450647)
[PASS] test_vaultWithdraw_nonWhitelistedAccount_reverts() (gas: 27596)
[PASS] test_vaultWithdraw_nonWhitelistedVault_reverts() (gas: 30819)
[PASS] test_vaultWithdraw_success() (gas: 277006)
[PASS] test_vaultWithdraw_zeroAmount_reverts() (gas: 27822)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 1.19ms (577.21µs CPU time)

Ran 7 tests for test/unit/withdrawals/WithdrawERC20.t.sol:WithdrawERC20Test
[PASS] test_withdraw_anyERC20_regardlessOfSupportedState() (gas: 1268268)
[PASS] test_withdraw_erc20_success() (gas: 374541)
[PASS] test_withdraw_multipleTokens_success() (gas: 676934)
[PASS] test_withdraw_native_reverts() (gas: 27580)
[PASS] test_withdraw_notWhitelistedAccount_reverts() (gas: 3761435)
[PASS] test_withdraw_notWhitelistedRecipient_reverts() (gas: 84103)
[PASS] test_withdraw_zeroAmount_reverts() (gas: 80958)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 1.36ms (478.17µs CPU time)

Ran 4 tests for test/unit/withdrawals/WithdrawNative.t.sol:WithdrawNativeTest
[PASS] test_withdrawNative_notWhitelistedAccount_reverts() (gas: 3756598)
[PASS] test_withdrawNative_notWhitelistedRecipient_reverts() (gas: 46934)
[PASS] test_withdrawNative_success() (gas: 337752)
[PASS] test_withdrawNative_zeroAmount_reverts() (gas: 33730)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 1.01ms (267.96µs CPU time)

Ran 9 tests for test/unit/perps-trading/PerpBuyOrders.t.sol:PerpBuyOrdersTest
[PASS] testFuzz_placeBuyPerpOrder_variousPricesAndSizes(uint64,uint64,bool) (runs: 256, : 329862, ~: 329862)
[PASS] test_placeBuyPerpOrder_multiplePerpOrders() (gas: 866715)
[PASS] test_placeBuyPerpOrder_reduceOnly() (gas: 361508)
[PASS] test_placeBuyPerpOrder_revertsForNonWhitelistedAccount() (gas: 25994)
[PASS] test_placeBuyPerpOrder_revertsForNonWhitelistedPerp() (gas: 34525)
[PASS] test_placeBuyPerpOrder_revertsForZeroPrice() (gas: 26244)
[PASS] test_placeBuyPerpOrder_revertsForZeroSize() (gas: 26295)
[PASS] test_placeBuyPerpOrder_standard() (gas: 322985)
[PASS] test_zeroIndex_placeBuyOrder() (gas: 324365)
Suite result: ok. 9 passed; 0 failed; 0 skipped; finished in 46.13ms (44.58ms CPU time)

Ran 8 tests for test/unit/staking/Delegation.t.sol:DelegationTest
[PASS] testFuzz_delegate_variousAmounts(uint64) (runs: 256, : 250969, ~: 251047)
[PASS] test_delegate_maxValues() (gas: 280496)
[PASS] test_delegate_multipleAmounts_sameValidator() (gas: 612202)
[PASS] test_delegate_multipleValidators() (gas: 699992)
[PASS] test_delegate_notWhitelistedAccount_reverts() (gas: 3690020)
[PASS] test_delegate_notWhitelistedValidator_reverts() (gas: 28692)
[PASS] test_delegate_whitelistedValidator_success() (gas: 251510)
[PASS] test_delegate_zeroAmount_reverts() (gas: 27709)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 15.77ms (15.09ms CPU time)

Ran 7 tests for test/unit/bridging/ForwardERC20.t.sol:ForwardERC20Test
[PASS] testFuzz_forward_variousAmounts(uint256) (runs: 256, : 385010, ~: 385010)
[PASS] test_forward_callsTransferERC20WithCorrectParams() (gas: 384234)
[PASS] test_forward_multipleTokensWithDifferentAmounts() (gas: 694243)
[PASS] test_forward_nativeTokenReverts() (gas: 34590)
[PASS] test_forward_revertsForNonWhitelistedAccount() (gas: 27732)
[PASS] test_forward_revertsForUnregisteredToken() (gas: 37939)
[PASS] test_forward_zeroAmount_reverts() (gas: 27828)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 28.12ms (26.31ms CPU time)

Ran 10 tests for test/unit/perps-trading/PerpSellOrders.t.sol:PerpSellOrdersTest
[PASS] testFuzz_placeSellPerpOrder_variousPricesAndSizes(uint64,uint64,bool) (runs: 256, : 334946, ~: 339766)
[PASS] test_placeSellPerpOrder_mixedBuyAndSellOrders() (gas: 555116)
[PASS] test_placeSellPerpOrder_multiplePerpOrders() (gas: 866640)
[PASS] test_placeSellPerpOrder_reduceOnly() (gas: 361484)
[PASS] test_placeSellPerpOrder_revertsForNonWhitelistedAccount() (gas: 25948)
[PASS] test_placeSellPerpOrder_revertsForNonWhitelistedPerp() (gas: 34500)
```

```
[PASS] test_placeSellPerpOrder_revertsForZeroPrice() (gas: 26176)
[PASS] test_placeSellPerpOrder_revertsForZeroSize() (gas: 26248)
[PASS] test_placeSellPerpOrder_standard() (gas: 323027)
[PASS] test_zeroIndex_placeSellOrder() (gas: 586244)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 48.29ms (47.83ms CPU time)

Ran 8 tests for test/unit/spot-trading/SpotOrderCancellation.t.sol:SpotOrderCancellationTest
[PASS] test_cancelSpotOrderByOid_callsCancelOrderByOidWithCorrectParams() (gas: 1203957)
[PASS] test_cancelSpotOrderByOid_multipleCancellations() (gas: 1555407)
[PASS] test_cancelSpotOrderByOid_notWhitelistedSpot_native_reverts() (gas: 58521)
[PASS] test_cancelSpotOrderByOid_notWhitelistedSpot_reverts() (gas: 949123)
[PASS] test_cancelSpotOrderByOid_revertsForNonWhitelistedAccount() (gas: 945751)
[PASS] test_cancelSpotOrderByOid_variousOids() (gas: 1734633)
[PASS] test_cancelSpotOrderByOid_verifyCallTracking() (gas: 1208154)
[PASS] test_cancelSpotOrderByOid_zeroOid_reverts() (gas: 984939)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 1.59ms (907.37µs CPU time)

Ran 5 tests for test/unit/spot-trading/SpotBuyOrders.t.sol:SpotBuyOrdersTest
[PASS] testFuzz_placeBuySpotOrder_variousPricesAndSizes(uint64,uint64) (runs: 256, : 344764, ~: 344764)
[PASS] test_placeBuySpotOrder_callsLimitOrderWithCorrectParams() (gas: 346376)
[PASS] test_placeBuySpotOrder_revertsForNonWhitelistedAccount() (gas: 27818)
[PASS] test_placeBuySpotOrder_revertsForNonWhitelistedToken() (gas: 955550)
[PASS] test_placeBuySpotOrder_revertsForUnregisteredToken() (gas: 38074)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 51.39ms (50.62ms CPU time)

Ran 15 tests for test/unit/spot-trading/SpotPriceBounds.t.sol:SpotPriceBoundsTest
[PASS] test_setPriceBounds_allZeroValues_reverts() (gas: 23891)
[PASS] test_setPriceBounds_differentBuyAndSell_success() (gas: 66295)
[PASS] test_setPriceBounds_invalidBuyBounds_reverts() (gas: 23523)
[PASS] test_setPriceBounds_invalidSellBounds_reverts() (gas: 23483)
[PASS] test_setPriceBounds_multipleTokens_success() (gas: 1023884)
[PASS] test_setPriceBounds_spotNotConfigured_reverts() (gas: 946956)
[PASS] test_setPriceBounds_success() (gas: 69881)
[PASS] test_setPriceBounds_tokenNotConfigured() (gas: 921632)
[PASS] test_setPriceBounds_updateExisting_success() (gas: 78788)
[PASS] test_setPriceBounds_zeroMinBuyPrice_reverts() (gas: 23473)
[PASS] test_setPriceBounds_zeroMinSellPrice_reverts() (gas: 23544)
[PASS] test_unsetPriceBounds_blocksTrading() (gas: 415617)
[PASS] test_unsetPriceBounds_notSet_reverts() (gas: 25648)
[PASS] test_unsetPriceBounds_success() (gas: 61559)
[PASS] test_unsetPriceBounds_tokenNotConfigured() (gas: 935756)
Suite result: ok. 15 passed; 0 failed; 0 skipped; finished in 3.48ms (1.78ms CPU time)

Ran 6 tests for test/unit/staking/StakingDeposits.t.sol:StakingDepositsTest
[PASS] test_depositStaking_maxAmount_success() (gas: 214863)
[PASS] test_depositStaking_minAmount_success() (gas: 214819)
[PASS] test_depositStaking_notWhitelistedAccount_reverts() (gas: 3687541)
[PASS] test_depositStaking_validAmount_success() (gas: 214907)
[PASS] test_depositStaking_variousAmounts() (gas: 814537)
[PASS] test_depositStaking_zeroAmount_reverts() (gas: 25255)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 1.06ms (414.46µs CPU time)

Ran 1 test for test/integration/flows/StakingFlow.t.sol:StakingFlowTest
[PASS] test_completeStakingFlow() (gas: 549002)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 3.09ms (762.71µs CPU time)

Ran 1 test for test/integration/scenarios/PerpDeprecation.t.sol:PerpDeprecationTest
[PASS] test_perpDeprecationFlow() (gas: 589534)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 11.51ms (2.21ms CPU time)

Ran 10 tests for test/unit/spot-trading/SpotSellOrders.t.sol:SpotSellOrdersTest
[PASS] testFuzz_placeSellSpotOrder_variousPricesAndSizes(uint64,uint64) (runs: 256, : 351018, ~: 344877)
[PASS] test_placeSellSpotOrder_callsLimitOrderWithCorrectParams() (gas: 349900)
[PASS] test_placeSellSpotOrder_multipleOrders() (gas: 610684)
[PASS] test_placeSellSpotOrder_priceAboveMax_reverts() (gas: 44530)
[PASS] test_placeSellSpotOrder_priceBelowMin_reverts() (gas: 44464)
[PASS] test_placeSellSpotOrder_revertsForNonWhitelistedAccount() (gas: 27928)
[PASS] test_placeSellSpotOrder_revertsForNonWhitelistedToken() (gas: 994377)
[PASS] test_placeSellSpotOrder_revertsForUnregisteredToken() (gas: 38184)
[PASS] test_placeSellSpotOrder_zeroPrice_reverts() (gas: 28199)
[PASS] test_placeSellSpotOrder_zeroSize_reverts() (gas: 28205)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 33.81ms (33.04ms CPU time)
```

```
Ran 9 tests for test/unit/perps-trading/PerpOrderCancellation.t.sol:PerpOrderCancellationTest
[PASS] testFuzz_cancelPerpOrderByOid_variousPerpsAndOids(uint8,uint64) (runs: 256, : 247400, ~: 252537)
[PASS] test_cancelPerpOrderByOid_callsCancelOrderByOidWithCorrectParams() (gas: 254038)
[PASS] test_cancelPerpOrderByOid_multipleCancellations() (gas: 608376)
[PASS] test_cancelPerpOrderByOid_revertsForNonWhitelistedAccount() (gas: 25376)
[PASS] test_cancelPerpOrderByOid_revertsForNonWhitelistedPerp() (gas: 33625)
[PASS] test_cancelPerpOrderByOid_variousOids() (gas: 735758)
[PASS] test_cancelPerpOrderByOid_verifyCallTracking() (gas: 255292)
[PASS] test_cancelPerpOrderByOid_zeroOid_reverts() (gas: 25627)
[PASS] test_zeroIndex_cancelOrder() (gas: 391188)
Suite result: ok. 9 passed; 0 failed; 0 skipped; finished in 13.20ms (13.19ms CPU time)

Ran 10 tests for test/unit/whitelists/SpotWhitelist.t.sol:SpotWhitelistTest
[PASS] test_getSpotWhitelist_native() (gas: 3773613)
[PASS] test_setSpotWhitelist_add_success() (gas: 72814)
[PASS] test_setSpotWhitelist_duplicate_idempotent() (gas: 80682)
[PASS] test_setSpotWhitelist_multipleTokens() (gas: 386835)
[PASS] test_setSpotWhitelist_remove_success() (gas: 69618)
[PASS] test_setSpotWhitelist_revertsForNonWhitelistedAccount() (gas: 27528)
[PASS] test_setSpotWhitelist_withConfig() (gas: 3931644)
[PASS] test_setSpotWhitelist_zeroTokenAddress_reverts() (gas: 35665)
[PASS] test_spotWhitelist_accountIsolation() (gas: 3761673)
[PASS] test_spotWhitelist_removeAndReAdd() (gas: 113588)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 4.80ms (1.64ms CPU time)

Ran 1 test for test/integration/flows/SpotFlow.t.sol:SpotFlowTest
[PASS] test_spotOrderFlow() (gas: 571312)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 13.44ms (2.40ms CPU time)

Ran 6 tests for test/unit/transfers/HyperCoreAccountNativeTransfers.t.sol:HyperCoreAccountNativeTransfersTest
[PASS] testFuzz_transferNative(uint256,uint256) (runs: 256, : 63165, ~: 63165)
[PASS] testFuzz_transferNative_multipleRecipients(uint256,uint8) (runs: 256, : 267486, ~: 251841)
[PASS] testFuzz_transferNative_partialAmount(uint256,uint256,uint256) (runs: 256, : 63292, ~: 63292)
[PASS] testFuzz_transferNative_revertsInsufficientBalance(uint256,uint256) (runs: 256, : 29893, ~: 29854)
[PASS] testFuzz_transferNative_revertsNonPayableRecipient(uint256) (runs: 256, : 78581, ~: 78581)
[PASS] testFuzz_transferNative_toContract(uint256) (runs: 256, : 79990, ~: 79990)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 99.93ms (156.92ms CPU time)

Ran 4 tests for test/unit/transfers/HyperCoreAccountERC20Transfers.t.sol:HyperCoreAccountERC20TransfersTest
[PASS] testFuzz_transferToken(uint256,address) (runs: 256, : 977686, ~: 977686)
[PASS] testFuzz_transferToken_multipleRecipients(uint256,uint8) (runs: 256, : 1142234, ~: 1127634)
[PASS] testFuzz_transferToken_partialAmount(uint256,uint256,address) (runs: 256, : 993826, ~: 993826)
[PASS] testFuzz_transferToken_revertsInsufficientBalance(uint256,uint256) (runs: 256, : 964869, ~: 966310)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 115.87ms (176.88ms CPU time)

Ran 38 tests for test/unit/access/TradeStakeManagerAccess.t.sol:TradeStakeManagerAccessTest
[PASS] test_bridgeFromL1_onlyOperator() (gas: 192500)
[PASS] test_cancelPerpOrderByOid_onlyOperator() (gas: 150571)
[PASS] test_cancelSpotOrderByOid_onlyOperator() (gas: 100791)
[PASS] test_delegate_onlyOperator() (gas: 121062)
[PASS] test_depositStaking_onlyOperator() (gas: 85488)
[PASS] test_forwardNative_onlyOperator() (gas: 65935)
[PASS] test_forward_onlyOperator() (gas: 1052055)
[PASS] test_initialization_grantsOperatorRoleToGovernor() (gas: 7030611)
[PASS] test_initialization_roleTransferPattern() (gas: 7095298)
[PASS] test_initialization_rolesSetCorrectly() (gas: 7075777)
[PASS] test_initialize_implementation_reverts() (gas: 16098)
[PASS] test_initialize_onlyCallableOnce() (gas: 47101)
[PASS] test_multipleRoles_independence() (gas: 191913)
[PASS] test_perpToSpot_onlyOperator() (gas: 114998)
[PASS] test_placeBuyPerpOrder_onlyOperator() (gas: 221508)
[PASS] test_placeBuySpotOrder_onlyOperator() (gas: 180999)
[PASS] test_placeSellPerpOrder_onlyOperator() (gas: 221529)
[PASS] test_placeSellSpotOrder_onlyOperator() (gas: 181293)
[PASS] test_publicGetters_anyoneCanCall() (gas: 98405)
[PASS] test_setAccountWhitelist_onlyGovernor() (gas: 52625)
[PASS] test_setPerpWhitelist_onlyGovernor() (gas: 108244)
[PASS] test_setPriceBounds_onlyGovernor() (gas: 68961)
[PASS] test_setSpotWhitelist_onlyGovernor() (gas: 122366)
[PASS] test_setTransferWhitelist_add_onlyGovernor() (gas: 65168)
[PASS] test_setTransferWhitelist_remove_onlyGovernor() (gas: 50267)
[PASS] test_setValidatorWhitelist_add_onlyGovernor() (gas: 78880)
[PASS] test_setValidatorWhitelist_remove_onlyGovernor() (gas: 50096)
```

```
[PASS] test_spotSendUSDC_onlyOperator() (gas: 174846)
[PASS] test_spotSend_onlyOperator() (gas: 162148)
[PASS] test_spotToPerp_onlyOperator() (gas: 115018)
[PASS] test_undelegate_onlyOperator() (gas: 121309)
[PASS] test_unsetPriceBounds_onlyGovernor() (gas: 66696)
[PASS] test_upgrade_notDefaultAdmin_reverts() (gas: 5960509)
[PASS] test_upgrade_notGovernor_reverts() (gas: 5978016)
[PASS] test_upgrade_onlyProxyAdmin_success() (gas: 6002054)
[PASS] test_withdrawNative_onlyOperator() (gas: 106275)
[PASS] test_withdrawStaking_onlyOperator() (gas: 85536)
[PASS] test_withdraw_onlyOperator() (gas: 1084532)
Suite result: ok. 38 passed; 0 failed; 0 skipped; finished in 156.17ms (17.64ms CPU time)

Ran 8 tests for test/unit/corewriter-account/CoreWriterActionEncoding.t.sol:HyperCoreAccountCoreWriterTest
[PASS] testFuzz_cancelOrderByOid(uint32,uint64) (runs: 256, : 92162, ~: 92162)
[PASS] testFuzz_limitOrder(uint32,bool,uint64,uint64,bool,uint8,uint128) (runs: 256, : 168584, ~: 169035)
[PASS] testFuzz_spotSend(address,uint64,uint64) (runs: 256, : 106810, ~: 106810)
[PASS] testFuzz_stakingDeposit(uint64) (runs: 256, : 77858, ~: 77858)
[PASS] testFuzz_stakingWithdraw(uint64) (runs: 256, : 77792, ~: 77792)
[PASS] testFuzz_tokenDelegate(address,uint64,bool) (runs: 256, : 106856, ~: 106856)
[PASS] testFuzz_usdClassTransfer(uint64,bool) (runs: 256, : 92126, ~: 92126)
[PASS] testFuzz_vaultTransfer(address,bool,uint64) (runs: 256, : 106810, ~: 106810)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 156.18ms (403.95ms CPU time)

Ran 8 tests for test/unit/corewriter-account/HyperCoreAccountEvents.t.sol:HyperCoreAccountEventsTest
[PASS] testFuzz_cancelOrderByOid_emitsEvent(uint32,uint64) (runs: 256, : 57639, ~: 57639)
[PASS] testFuzz_limitOrder_emitsEvent(uint32,bool,uint64,uint64,bool,uint8,uint128) (runs: 256, : 71952, ~: 72358)
[PASS] testFuzz_spotSend_emitsEvent(address,uint64,uint64) (runs: 256, : 59853, ~: 59853)
[PASS] testFuzz_stakingDeposit_emitsEvent(uint64) (runs: 256, : 55838, ~: 55838)
[PASS] testFuzz_stakingWithdraw_emitsEvent(uint64) (runs: 256, : 55819, ~: 55819)
[PASS] testFuzz_tokenDelegate_emitsEvent(address,uint64,bool) (runs: 256, : 59876, ~: 59876)
[PASS] testFuzz_usdClassTransfer_emitsEvent(uint64,bool) (runs: 256, : 57624, ~: 57624)
[PASS] testFuzz_vaultTransfer_emitsEvent(address,bool,uint64) (runs: 256, : 59854, ~: 59854)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 144.39ms (222.21ms CPU time)

Ran 60 test suites in 244.94ms (1.22s CPU time): 439 tests passed, 0 failed, 0 skipped (439 total tests)
```

11.3 Notes on the Test Suite

The provided test suite demonstrates a well-structured and comprehensive approach to validating the functionality of the Trade and Stake Managers and their associated components. The tests are logically organized into discrete units, with additional scenario-based tests that verify complex system interactions.

The suite covers a wide range of cases, including:

- **Core logic validation** – order execution, liquidation, staking, delegation/undelegation, vault operations, and bridging.
- **Access control enforcement** – ensuring that only authorized roles (e.g., owner, admin, proxy admin) can invoke privileged functions.
- **Failure and edge cases** – appropriate handling of invalid inputs, boundary conditions, and unauthorized actions.
- **Integration flows** – cross-module operations such as migrations, proxy upgrades, and withdrawals.
- **Fuzz testing** – applied to functions with large input domains (e.g., delegation, bridging, order placement) to explore diverse state transitions.

Overall, the test coverage is strong. The suite provides assurance that both positive and negative execution paths are exercised, and fuzzing contributes additional confidence against unexpected behaviors. The test suite as provided is of high quality and offers strong confidence in the correctness and robustness of the system.