Quantstamp

Berachain - BEX

Executive Summary

This audit report was prepared by Quantstamp, the leader in blockchain security.

Type	DeFi
Timeline	2025-01-08 through 2025-01-23
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	README.md Berachain BEX Documentation Resource for BEX Auditors
Diff/Fork information	The BEX repository was forked from the Balancer V2 repository at commit `36d2823`. The BEX deployment repository was forked from Balancer's deployment configuration repository at commit `d81a658`.
Source Code	• berachain/balancer-v2-monorepo #c28a86b • berachain/balancer-deployments #e0ce02d • berachain/balancer-v2-vault-v2 #34d3082 • berachain/balancer-v2-weighted-pool-v5 #5e15464
Auditors	• Joseph Xu Technical R&D Advisor • Andrei Stefan Auditing Engineer

Documentation quality	Medium
Test quality	High
Total Findings	5 Fixed: 2 Acknowledged: 2 Mitigated: 1
High severity findings	0
Medium severity findings	1 Fixed: 1
Low severity findings	1 Fixed: 1
Undetermined severity findings	3 Acknowledged: 2 Mitigated: 1
Informational findings	0

Summary of Findings

Final Report (2025-01-23): Quantstamp has reviewed response from the Berachain BEX team and an additional commit hash `191939d` for the `balancer-v2-monorepo` containing fixes. The commit contains a major update to the price oracle contract, as well as an earlier commit that fixes the test suite. As of this commit, all of the salient issues identified in the Initial Report has either been Fixed or Mitigated. One remaining issue that is Acknowledged is due to issues from Balancer V2 carrying over to BEX, which is difficult to address completely. Berachain BEX team is aware of this issue and intends to operate BEX with proper safeguards.

Initial Report (2025-01-17): Quantstamp has conducted a diff audit of BEX, which is Berachain's native AMM that has been forked from Balancer V2. Berachain BEX team has made the following modifications on top of Balancer V2:

1. A new feature to create a liquidity pool and enter the liquidity pool in the same transaction (`PoolCreationHelper.sol` contract).
2. A new contract to act as an oracle for HONEY-USDC-PYUSD stablecoin tri-pool (`SpotPriceOracle.sol` contract).
3. Updating protocol constants to allow up to 5 years in pause windows.
4. Updating protocol constants to discourage/prevent the use of the flash loan feature.
5. Updating the mechanisms for fee withdrawal to match the Proof-of-Liquidity mechanisms on Berachain.

In addition to auditing the code diff in BEX, Quantstamp has also reviewed some of the deployment scripts for deploying BEX onto the Berachain mainnet.

The auditors, alongside the Berachain BEX team, have identified 1 Medium, 1 Low, and 3 Undetermined severity issues. Most of these issues stem from the lack of information and context on the intended usage of new smart contracts and deployment procedures. While these issues are unlikely to cause severe consequences for BEX, the auditors strongly recommend addressing these issues through better documentation and more rigorous operational procedures so that unintended errors can be avoided in production.

ID	DESCRIPTION	SEVERITY	STATUS
BEX-1	Incorrect Asset Price Calculation in the Oracle's Internal Function	● Medium	Fixed
BEX-2	Potential Placeholder Values for Important Addresses and Variables in Deployment Scripts	● Low	Fixed
BEX-3	Oracle Functions for Returning Price Feed May Be Incomplete	● Undetermined	Acknowledged
BEX-4	Known Issues in Balancer V2 Are Also Present in BEX	● Undetermined	Acknowledged
BEX-5	Spot Price Calculation Is Vulnerable to Manipulations	● Undetermined	Mitigated

Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Disclaimer

Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

1. Code review that includes the following
   1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Scope

The scope of this audit is limited to new and modified files from the original Balancer V2 smart contract and deployment script repositories. The exact files and directories included in the audit scope are listed below.

Files Included

Within the `berachain/balancer-v2-monorepo` repository:

- `/pkg/interfaces/contracts/pool-stable/IComposableStablePoolFactoryCreateV6.sol`
- `/pkg/interfaces/contracts/standalone-utils/IPriceOracle.sol`
- `/pkg/interfaces/contracts/standalone-utils/IProtocolFeesWithdrawer.sol`
- `/pkg/standalone-utils/contracts/PoolCreationHelper.sol`
- `/pkg/standalone-utils/contracts/ProtocolFeePercentagesProvider.sol`
- `/pkg/standalone-utils/contracts/ProtocolFeesWithdrawer.sol`
- `/pkg/standalone-utils/contracts/SpotPriceOracle.sol`

Within the `berachain/balancer-deployments` repository:

- `/src/deploymentConfig.ts`
- `/task/00000000-tokens/*`
- `/tasks/20210418-authorizer/*`
- `/tasks/20220721-balancer-queries/*`
- `/tasks/20231031-batch-relayer-v6/*`
- `/tasks/20240223-composable-stable-pool-v6/*`
- `/tasks/20210418-vault/*`
- `/tasks/20241119-vault-v2/*`
- `/tasks/20230320-weighted-pool-v4/*`
- `/tasks/20210418-vault/*`
- `/tasks/20241121-weighted-pool-v5/*`
- `/tasks/20241119-pool-creation-helper/*`
- `/tasks/20241125-protocol-fee-percentages-provider-v2/*`
- `/tasks/20241025-protocol-fees-withdrawer-v2/*`
- `/hardhat.config.ts`

Findings

BEX-1
Incorrect Asset Price Calculation in the Oracle's Internal Function

● Medium ⓘ Fixed

Update
This issue is fixed as of commit `191939d`.

File(s) affected: `balancer-v2-monorepo/pkg/standalone-utils/contracts/SpotPriceOracle.sol`

Description: The function `SpotPriceOracle._getPriceOutOfIn()` does not return the price of the output asset with respect to the input asset, but instead returns the default amount of input asset (1 USDC or 1 PYUSD). This is due to the calculation of return value `price = scaledAmountIn.divDown(scaledAmountOut)` in L132 incorrectly using the variables `scaledAmountIn` and `scaledAmountOut` as the dividend and the divisor respectively. `scaledAmountOut` should be the dividend and `scaledAmountIn` should be the divisor.

While this would be a critical issue in any other circumstances, the effect is somewhat reduced in this context because the oracle is used to return the relative price of two stablecoin tokens with the same decimal units.

Recommendation: Fix L132 to `price = scaledAmountOut.divDown(scaledAmountIn);`

BEX-2
Potential Placeholder Values for Important Addresses and Variables in Deployment Scripts

● Low ⓘ Fixed

Update
Berachain BEX team has confirmed the correctness of the addresses and values. In addition, the BEX team has provided an internal document that details the deployment procedures and checks to be performed before deployment.

File(s) affected: `balancer-deployments/tasks/00000000-tokens/output/berachain.json`, `balancer-deployments/tasks/20210418-authorizer/input.ts`, `balancer-deployments/tasks/20231031-batch-relayer-v6/input.ts`, `balancer-deployments/tasks/20241025-protocol-fees-withdrawer-v2/input.ts`

Description: The deployment scripts still contain values that may be placeholders for important addresses and variable values. Failure to update placeholder values before deployment can result in failed deployment or deployment with incorrect parameters.

Specifically:

- `/tasks/00000000-tokens/output/berachain.json` - the WETH token address on the Berachain mainnet currently has the value `0x6969696969696969696969696969696969696969`.
- `/tasks/20210418-authorizer/input.ts` - admin address on the Berachain mainnet is not available and the value is 'add-admin-address-here'.
- `/tasks/20231031-batch-relayer-v6/input.ts` - for the `cartio` and `berachain` networks, the address for field `wstETH` is `ZERO_ADDRESS`, the address for `BalancerMinter` is `ZERO_ADDRESS`, and the field `CanCallUserCheckpoint` is `false`.
- `/tasks/20241025-protocol-fees-withdrawer-v2/input.ts` - for the `berachain` network, the `polFeeCollector` and `feeReceiver` addresses are both unavailable; the values are 'pol-fee-collector-address' and 'fee-receiver-address' respectively.

Recommendation: Replace the placeholder values to addresses and values actually used in the deployment. If the proper addresses or values are not known at the moment, document the exact deployment procedures and checks that need to be performed before deployment.

BEX-3
Oracle Functions for Returning Price Feed May Be Incomplete

● Undetermined ⓘ Acknowledged

Update
Berachain BEX team has provided additional context that the oracle contract is intended to be a fallback oracle in case there is an issue with the primary oracle (Pyth). Since the oracle is using the live pool balances to compute the price feed, the `block.timestamp` being the price feed timestamp is not an issue. The Berachain BEX team also added comments in commit `191939d` indicating that `getPriceUnsafe()` and `getPriceNoOlderThan()` will have the same return value as `getPrice()`.

File(s) affected: `balancer-v2-monorepo/pkg/interfaces/contracts/standalone-utils/IPriceOracle.sol`, `balancer-v2-monorepo/pkg/standalone-utils/contracts/SpotPriceOracle.sol`

Description: The oracle contract `SpotPriceOracle.sol` has three functions that are used to return price feed:

- `getPrice()`
- `getPriceUnsafe()`
- `getPriceNoOlderThan()`

All three functions return the same value, with only the `getPriceNoOlderThan()` function having an additional check preventing the oracle from serving any price older than `age == 0`. All three functions return `block.timestamp` as the timestamp of the price feed. There are no documentation or code comments on the specs of these functions, and it appears that the latter two functions are currently incomplete.

Recommendation: Provide the intended specification for the oracle price feed.

BEX-4
Known Issues in Balancer V2 Are Also Present in BEX

● Undetermined ⓘ Acknowledged

Update
Berachain BEX team is aware of issues in Balancer V2. There are documentations and also comments within the repository that provide information on the proper usage of the protocol/smart contracts in light of these issues.

File(s) affected: `balancer-v2-monorepo/pkg/standalone-utils/contracts/PoolCreationHelper.sol`, `balancer-v2-monorepo/pkg/standalone-utils/contracts/ProtocolFeesWithdrawer.sol`

Description: Balancer V2 has a known issues where the pools do not support non-standard/double-entry tokens. This issue affect new features such as the `PoolCreationHelper.sol` contract or modified features such as the `ProtocolFeesWithdrawer.sol` such that they also do not support non-standard/double-entry tokens.

More generally, due to the nature of BEX as a Balancer fork with fairly minimal modifications, other issues present in Balancer V2 also carry over and affect BEX in mostly similar ways.

Recommendation: Write documentations and references that highlight known issues in Balancer V2 so that BEX do not accidentally create conditions where these issues cause significant vulnerabilities.

BEX-5
Spot Price Calculation Is Vulnerable to Manipulations

● Undetermined ⓘ Mitigated

Update
Berachain BEX team has provided additional context that the oracle contract is intended to be a fallback oracle in case there is an issue with the primary oracle (Pyth). The Berachain BEX team also added comments in commit `191939d` indicating that the oracle spot price

may be manipulated and should not be used directly by external dApps.

File(s) affected: `balancer-v2-monorepo/pkg/standalone-utils/contracts/SpotPriceOracle.sol`

Description: The oracle is vulnerable to asset price manipulation because the internal function used to compute the asset price `_getAssetPrice()` reads the token balances from a stablecoin pool and uses these values directly for its spot price calculations. The pool balances can be easily manipulated by flash loans or by making repetitive trades in a certain direction to move the price. External protocols and dApps referencing this oracle may suffer from spot price that deviate significantly from the prevailing market price.

Exploit Scenario:
1. The oracle calculates asset price based on the pool composition of the HONEY-USDC-PYUSD tri-pool.
2. The attacker borrows a large amount of one pool token using flash loan.
3. The attacker executes a large swap in the HONEY-USDC-PYUSD tri-pool.
4. The oracle reads the manipulated pool balances from the flash loan trade and returns a price that is significantly deviated from the market.
5. The attacker executes transactions within the Berachain ecosystem to extract value from dApps that rely on the oracle, which is now returning a manipulated price.
6. The attacker profits, unwinds the trade that caused the pool balance dislocation, and repays the flash loan.

Recommendation: Allow the oracle to return a TWAP price of the assets. However, even TWAP prices can be manipulated in certain circumstances. Therefore it is important for the stablecoin pool used in the spot price calculation to have enough liquidity that is being rebalanced frequently.

For more research on this issue, see the following articles:
1. Balancer V2 documentation on 'Oracles (deprecated)'
2. ChainSecurity article on oracle manipulation

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Files

- `3f8...757 ./pkg/interfaces/contracts/standalone-utils/IPriceOracle.sol`
- `af1...99b ./pkg/interfaces/contracts/standalone-utils/IProtocolFeesWithdrawer.sol`
- `bf5...2be ./pkg/interfaces/contracts/pool-stable/IComposableStablePoolFactoryCreateV6.sol`
- `13a...2ec ./pkg/standalone-utils/contracts/SpotPriceOracle.sol`
- `dc1...8c2 ./pkg/standalone-utils/contracts/ProtocolFeesWithdrawer.sol`
- `4fb...9c8 ./pkg/standalone-utils/contracts/ProtocolFeePercentagesProvider.sol`
- `4f6...d52 ./pkg/standalone-utils/contracts/PoolCreationHelper.sol`

Test Suite Results

There were several failing tests in the Initial Audit commit `c28a86b`. This was addressed in PR #15.

All tests are passing in the `@balancer-labs/v2-standalone-utils` package as of commit `2cdbdf6`.

```
AaveWrapping
  wrapAaveDynamicToken
    when caller != sender and sender != relayer
      ✔ reverts
    from underlying tokens
      sender = user
        using immediate amounts
          ✔ pulls tokens if needed
          ✔ approves static token to spend dynamic tokens
          ✔ deposits dynamic tokens
          ✔ stores wrap output as chained reference
        using chained references
          ✔ pulls tokens if needed
          ✔ approves static token to spend dynamic tokens
          ✔ deposits dynamic tokens
          ✔ stores wrap output as chained reference
      sender = relayer
        using immediate amounts
          ✔ pulls tokens if needed
          ✔ approves static token to spend dynamic tokens
          ✔ deposits dynamic tokens
          ✔ stores wrap output as chained reference
        using chained references
          ✔ pulls tokens if needed
          ✔ approves static token to spend dynamic tokens
          ✔ deposits dynamic tokens
          ✔ stores wrap output as chained reference
    from dynamic aTokens
      sender = user
        using immediate amounts
          ✔ pulls tokens if needed
          ✔ approves static token to spend dynamic tokens
          ✔ deposits dynamic tokens
          ✔ stores wrap output as chained reference
        using chained references
          ✔ pulls tokens if needed
          ✔ approves static token to spend dynamic tokens
          ✔ deposits dynamic tokens
          ✔ stores wrap output as chained reference
      sender = relayer
        using immediate amounts
          ✔ pulls tokens if needed
          ✔ approves static token to spend dynamic tokens
          ✔ deposits dynamic tokens
          ✔ stores wrap output as chained reference
        using chained references
          ✔ pulls tokens if needed
          ✔ approves static token to spend dynamic tokens
          ✔ deposits dynamic tokens
          ✔ stores wrap output as chained reference
  unwrapAaveDynamicToken
    when caller != sender and sender != relayer
      ✔ reverts
    to underlying tokens
      sender = user
        using immediate amounts
          ✔ pulls static tokens if needed
          ✔ withdraws dynamic tokens
          ✔ stores unwrap output as chained reference
        using chained references
          ✔ pulls static tokens if needed
          ✔ withdraws dynamic tokens
          ✔ stores unwrap output as chained reference
      sender = relayer
```

```
        using immediate amounts
          ✔ pulls static tokens if needed
          ✔ withdraws dynamic tokens
          ✔ stores unwrap output as chained reference
        using chained references
          ✔ pulls static tokens if needed
          ✔ withdraws dynamic tokens
          ✔ stores unwrap output as chained reference
    to dynamic aTokens
      sender = user
        using immediate amounts
          ✔ pulls static tokens if needed
          ✔ withdraws dynamic tokens
          ✔ stores unwrap output as chained reference
        using chained references
          ✔ pulls static tokens if needed
          ✔ withdraws dynamic tokens
          ✔ stores unwrap output as chained reference
      sender = relayer
        using immediate amounts
          ✔ pulls static tokens if needed
          ✔ withdraws dynamic tokens
          ✔ stores unwrap output as chained reference
        using chained references
          ✔ pulls static tokens if needed
          ✔ withdraws dynamic tokens
          ✔ stores unwrap output as chained reference

BALTokenHolder
  ✔ returns the BAL address
  ✔ returns its name
  withdrawFunds
    when the caller is authorized
      ✔ sends funds to the recipient
    when the caller is not authorized
      ✔ reverts
  sweepTokens
    when the caller is authorized
      when the token is not BAL
        ✔ sends funds to the recipient
      when the token is BAL
        ✔ reverts
    when the caller is not authorized
      ✔ reverts

BALTokenHolderFactory
  ✔ returns the BAL address
  ✔ returns the address of the vault
  creation
    ✔ emits an event
    ✔ creates a holder with the same BAL and vault addresses
    ✔ creates a holder with name
    ✔ creates holders with unique action IDs
  is holder from factory
    ✔ returns true for holders created by the factory
    ✔ returns false for other addresses

BalancerQueries
  querySwap
    ✔ can query swap results
    ✔ bubbles up revert reasons
  queryBatchSwap
    ✔ can query batch swap results
    ✔ bubbles up revert reasons
  queryJoin
    ✔ can query join results
    ✔ bubbles up revert reasons
    when the pool is paused
      ✔ reverts
  queryExit
    ✔ can query exit results
    ✔ bubbles up revert reasons
    when the pool is paused
      ✔ reverts

  BaseRelayerLibrary
    relayer getters
      ✔ returns the library address
      ✔ returns the query library address
      ✔ returns the vault address
      ✔ returns the relayer version
    chained references
      ✔ identifies immediate amounts
      ✔ identifies chained references
      read and write
        when the reference is temporary
          ✔ reads uninitialized references as zero
          ✔ reads stored references
          ✔ writes replace old data
          ✔ stored data in independent slots
          ✔ peeks uninitialized references as zero
          ✔ peeks stored references
          ✔ peeks overwritten data
          ✔ peeks stored data in independent slots
          ✔ peeks same slot multiple times
          ✔ peeks and reads same slot
          ✔ clears data after reading
        when the reference is not temporary
          ✔ reads uninitialized references as zero
          ✔ reads stored references
          ✔ writes replace old data
          ✔ stored data in independent slots
          ✔ peeks uninitialized references as zero
          ✔ peeks stored references
          ✔ peeks overwritten data
          ✔ peeks stored data in independent slots
          ✔ peeks same slot multiple times
          ✔ peeks and reads same slot
          ✔ preserves data after reading
        when mixing temporary and read-only references
          ✔ writes the same slot (temporary write)
          ✔ writes the same slot (read-only write)
          ✔ reads the same written slot
          ✔ reads the same cleared slot
    multicall
      when msg.value is nonzero
        ✔ refunds the unused ETH
      setRelayerApproval
        when relayer is authorised by governance
          ✔ is payable
          when modifying its own approval
            ✔ sets the desired approval for the relayer to act for sender
            ✔ approval applies to later calls within the same multicall
          when modifying the approval for another relayer
            ✔ reverts when giving approval for another relayer
            ✔ correctly revokes approval for another relayer
        when relayer is not authorised by governance
          ✔ reverts
      peekChainedReferenceValue
        ✔ peeks chained reference
        ✔ is payable
    approve vault
      when using values as argument
        ✔ approves vault to use tokens
        ✔ is payable
      when using chained references as argument
        ✔ approves vault to use tokens
        ✔ is payable

  CompoundV2Wrapping
    primitives
      wrapCompoundV2
        sender = senderUser, recipient = relayer
          ✔ wraps with immediate amounts
          ✔ stores wrap output as chained reference
          ✔ wraps with chained references
        sender = senderUser, recipient = senderUser
          ✔ wraps with immediate amounts
          ✔ stores wrap output as chained reference
          ✔ wraps with chained references
        sender = relayer, recipient = relayer
          ✔ wraps with immediate amounts
          ✔ stores wrap output as chained reference
          ✔ wraps with chained references
        sender = relayer, recipient = senderUser
          ✔ wraps with immediate amounts
          ✔ stores wrap output as chained reference
          ✔ wraps with chained references
      unwrapCompoundV2
        sender = senderUser, recipient = relayer
          ✔ unwraps with immediate amounts
          ✔ stores unwrap output as chained reference
          ✔ unwraps with chained references
        sender = senderUser, recipient = senderUser
          ✔ unwraps with immediate amounts
          ✔ stores unwrap output as chained reference
          ✔ unwraps with chained references
        sender = relayer, recipient = relayer
          ✔ unwraps with immediate amounts
          ✔ stores unwrap output as chained reference
          ✔ unwraps with chained references
        sender = relayer, recipient = senderUser
          ✔ unwraps with immediate amounts
          ✔ stores unwrap output as chained reference
          ✔ unwraps with chained references
    complex actions
      swap
        swap using DAI as an input
          ✔ performs the given swap
          ✔ does not leave dust on the relayer
        swap using DAI as an output
          ✔ performs the given swap
          ✔ does not leave dust on the relayer
      batchSwap
        swap using DAI as an input
          ✔ performs the given swap
          ✔ does not leave dust on the relayer
        swap using DAI as an output
          ✔ performs the given swap
          ✔ does not leave dust on the relayer
      joinPool
        ✔ joins the pool
        ✔ does not take cDAI from the user
        ✔ does not leave dust on the relayer
      exitPool
        ✔ exits the pool
        ✔ BPT burned from the sender user
        ✔ DAI transfered to recipient user
        ✔ does not leave dust on the relayer

  EulerWrapping
    primitives
      wrap Euler
        sender = senderUser, recipient = relayer
          ✔ wraps with immediate amounts
      unwrap Euler
        sender = senderUser, recipient = relayer
          ✔ unwraps with immediate amounts
          ✔ stores unwrap output as chained reference
          ✔ unwraps with chained references
    complex actions
      swap
        swap using DAI as an input
          ✔ performs the given swap
          ✔ does not leave dust on the relayer
        swap using DAI as an output
          ✔ performs the given swap
          ✔ does not leave dust on the relayer
      batchswap
        swap using DAI as an input
          ✔ performs the given swap
          ✔ does not leave dust on the relayer
        swap using DAI as an output
          ✔ performs the given swap
          ✔ does not leave dust on the relayer
      joinPool
        ✔ joins the pool
        ✔ does not take eDAI from the user
        ✔ does not leave dust on the relayer
      exitPool
        ✔ exits the pool
        ✔ BPT burned from the sender user
        ✔ DAI transfered to recipient user
        ✔ does not leave dust on the relayer

  GaugeActions
    Liquidity gauge
      gaugeDeposit
        when using relayer library directly
          ✔ reverts
        when caller != sender and sender != relayer
          ✔ reverts
        when sender does not have enough BPT
          ✔ reverts
        when sender has enough BPT
          sender = user, recipient = user
            when depositing some of the tokens
              when using immediate amounts
                ✔ pulls BPT tokens from sender if necessary
                ✔ approves gauge to use relayer's BPT funds
                ✔ emits BPT transfer event from relayer to gauge if necessary
                ✔ transfers BPT tokens to gauge
                ✔ emits deposit event
                ✔ mints gauge tokens to recipient
                ✔ emits transfer event for minted gauge tokens
              when using chained references
                ✔ pulls BPT tokens from sender if necessary
                ✔ approves gauge to use relayer's BPT funds
                ✔ emits BPT transfer event from relayer to gauge if necessary
                ✔ transfers BPT tokens to gauge
                ✔ emits deposit event
                ✔ mints gauge tokens to recipient
                ✔ emits transfer event for minted gauge tokens
            when depositing all of the available tokens
              when using immediate amounts
                ✔ pulls BPT tokens from sender if necessary
                ✔ approves gauge to use relayer's BPT funds
                ✔ emits BPT transfer event from relayer to gauge if necessary
                ✔ transfers BPT tokens to gauge
                ✔ emits deposit event
                ✔ mints gauge tokens to recipient
                ✔ emits transfer event for minted gauge tokens
              when using chained references
                ✔ pulls BPT tokens from sender if necessary
                ✔ approves gauge to use relayer's BPT funds
                ✔ emits BPT transfer event from relayer to gauge if necessary
                ✔ transfers BPT tokens to gauge
                ✔ emits deposit event
                ✔ mints gauge tokens to recipient
                ✔ emits transfer event for minted gauge tokens
            when depositing 0 tokens
              when using immediate amounts
                ✔ pulls BPT tokens from sender if necessary
                ✔ approves gauge to use relayer's BPT funds
                ✔ emits BPT transfer event from relayer to gauge if necessary
                ✔ transfers BPT tokens to gauge
                ✔ emits deposit event
                ✔ mints gauge tokens to recipient
                ✔ emits transfer event for minted gauge tokens
              when using chained references
                ✔ pulls BPT tokens from sender if necessary
                ✔ approves gauge to use relayer's BPT funds
                ✔ emits BPT transfer event from relayer to gauge if necessary
                ✔ transfers BPT tokens to gauge
                ✔ emits deposit event
                ✔ mints gauge tokens to recipient
                ✔ emits transfer event for minted gauge tokens
          sender = user, recipient = relayer
            when depositing some of the tokens
              when using immediate amounts
                ✔ pulls BPT tokens from sender if necessary
                ✔ approves gauge to use relayer's BPT funds
                ✔ emits BPT transfer event from relayer to gauge if necessary
                ✔ transfers BPT tokens to gauge
                ✔ emits deposit event
                ✔ mints gauge tokens to recipient
                ✔ emits transfer event for minted gauge tokens
              when using chained references
                ✔ pulls BPT tokens from sender if necessary
                ✔ approves gauge to use relayer's BPT funds
                ✔ emits BPT transfer event from relayer to gauge if necessary
                ✔ transfers BPT tokens to gauge
                ✔ emits deposit event
                ✔ mints gauge tokens to recipient
                ✔ emits transfer event for minted gauge tokens
            when depositing all of the available tokens
              when using immediate amounts
                ✔ pulls BPT tokens from sender if necessary
                ✔ approves gauge to use relayer's BPT funds
                ✔ emits BPT transfer event from relayer to gauge if necessary
                ✔ transfers BPT tokens to gauge
                ✔ emits deposit event
                ✔ mints gauge tokens to recipient
                ✔ emits transfer event for minted gauge tokens
              when using chained references
                ✔ pulls BPT tokens from sender if necessary
                ✔ approves gauge to use relayer's BPT funds
                ✔ emits BPT transfer event from relayer to gauge if necessary
                ✔ transfers BPT tokens to gauge
                ✔ emits deposit event
                ✔ mints gauge tokens to recipient
                ✔ emits transfer event for minted gauge tokens
            when depositing 0 tokens
              when using immediate amounts
                ✔ pulls BPT tokens from sender if necessary
                ✔ approves gauge to use relayer's BPT funds
                ✔ emits BPT transfer event from relayer to gauge if necessary
                ✔ transfers BPT tokens to gauge
                ✔ emits deposit event
                ✔ mints gauge tokens to recipient
                ✔ emits transfer event for minted gauge tokens
              when using chained references
                ✔ pulls BPT tokens from sender if necessary
                ✔ approves gauge to use relayer's BPT funds
                ✔ emits BPT transfer event from relayer to gauge if necessary
                ✔ transfers BPT tokens to gauge
                ✔ emits deposit event
                ✔ mints gauge tokens to recipient
                ✔ emits transfer event for minted gauge tokens
          sender = relayer, recipient = user
            when depositing some of the tokens
              when using immediate amounts
                ✔ pulls BPT tokens from sender if necessary
                ✔ approves gauge to use relayer's BPT funds
                ✔ emits BPT transfer event from relayer to gauge if necessary
                ✔ transfers BPT tokens to gauge
                ✔ emits deposit event
                ✔ mints gauge tokens to recipient
                ✔ emits transfer event for minted gauge tokens
              when using chained references
```

```
        ✔ pulls BPT tokens from sender if necessary
        ✔ approves gauge to use relayer's BPT funds
        ✔ emits BPT transfer event from relayer to gauge if necessary
        ✔ transfers BPT tokens to gauge
        ✔ emits deposit event
        ✔ mints gauge tokens to recipient
        ✔ emits transfer event for minted gauge tokens
    when depositing all of the available tokens
      when using immediate amounts
        ✔ pulls BPT tokens from sender if necessary
        ✔ approves gauge to use relayer's BPT funds
        ✔ emits BPT transfer event from relayer to gauge if necessary
        ✔ transfers BPT tokens to gauge
        ✔ emits deposit event
        ✔ mints gauge tokens to recipient
        ✔ emits transfer event for minted gauge tokens
      when using chained references
        ✔ pulls BPT tokens from sender if necessary
        ✔ approves gauge to use relayer's BPT funds
        ✔ emits BPT transfer event from relayer to gauge if necessary
        ✔ transfers BPT tokens to gauge
        ✔ emits deposit event
        ✔ mints gauge tokens to recipient
        ✔ emits transfer event for minted gauge tokens
    when depositing 0 tokens
      when using immediate amounts
        ✔ pulls BPT tokens from sender if necessary
        ✔ approves gauge to use relayer's BPT funds
        ✔ emits BPT transfer event from relayer to gauge if necessary
        ✔ transfers BPT tokens to gauge
        ✔ emits deposit event
        ✔ mints gauge tokens to recipient
        ✔ emits transfer event for minted gauge tokens
      when using chained references
        ✔ pulls BPT tokens from sender if necessary
        ✔ approves gauge to use relayer's BPT funds
        ✔ emits BPT transfer event from relayer to gauge if necessary
        ✔ transfers BPT tokens to gauge
        ✔ emits deposit event
        ✔ mints gauge tokens to recipient
        ✔ emits transfer event for minted gauge tokens
  sender = relayer, recipient = relayer
    when depositing some of the tokens
      when using immediate amounts
        ✔ pulls BPT tokens from sender if necessary
        ✔ approves gauge to use relayer's BPT funds
        ✔ emits BPT transfer event from relayer to gauge if necessary
        ✔ transfers BPT tokens to gauge
        ✔ emits deposit event
        ✔ mints gauge tokens to recipient
        ✔ emits transfer event for minted gauge tokens
      when using chained references
        ✔ pulls BPT tokens from sender if necessary
        ✔ approves gauge to use relayer's BPT funds
        ✔ emits BPT transfer event from relayer to gauge if necessary
        ✔ transfers BPT tokens to gauge
        ✔ emits deposit event
        ✔ mints gauge tokens to recipient
        ✔ emits transfer event for minted gauge tokens
    when depositing all of the available tokens
      when using immediate amounts
        ✔ pulls BPT tokens from sender if necessary
        ✔ approves gauge to use relayer's BPT funds
        ✔ emits BPT transfer event from relayer to gauge if necessary
        ✔ transfers BPT tokens to gauge
        ✔ emits deposit event
        ✔ mints gauge tokens to recipient
        ✔ emits transfer event for minted gauge tokens
      when using chained references
        ✔ pulls BPT tokens from sender if necessary
        ✔ approves gauge to use relayer's BPT funds
        ✔ emits BPT transfer event from relayer to gauge if necessary
```

```
          ✔ transfers BPT tokens to gauge
          ✔ emits deposit event
          ✔ mints gauge tokens to recipient
          ✔ emits transfer event for minted gauge tokens
      when depositing 0 tokens
        when using immediate amounts
          ✔ pulls BPT tokens from sender if necessary
          ✔ approves gauge to use relayer's BPT funds
          ✔ emits BPT transfer event from relayer to gauge if necessary
          ✔ transfers BPT tokens to gauge
          ✔ emits deposit event
          ✔ mints gauge tokens to recipient
          ✔ emits transfer event for minted gauge tokens
        when using chained references
          ✔ pulls BPT tokens from sender if necessary
          ✔ approves gauge to use relayer's BPT funds
          ✔ emits BPT transfer event from relayer to gauge if necessary
          ✔ transfers BPT tokens to gauge
          ✔ emits deposit event
          ✔ mints gauge tokens to recipient
          ✔ emits transfer event for minted gauge tokens
gaugeWithdraw
  when using relayer library directly
    ✔ reverts
  when caller != sender and sender != relayer
    ✔ reverts
  when sender does not have enough gauge tokens
    ✔ reverts
  when sender has enough gauge tokens
    sender = user, recipient = user
      when withdrawing some of the tokens
        when using immediate amounts
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
        when using chained references
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
      when withdrawing all the available tokens
        when using immediate amounts
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
        when using chained references
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
      when withdrawing 0 tokens
        when using immediate amounts
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
        ✔ transfers BPT tokens to recipient
      when using chained references
        ✔ pulls gauge tokens from sender if necessary
        ✔ emits BPT transfer event from gauge to relayer if necessary
        ✔ emits withdraw event
        ✔ burns gauge tokens
        ✔ emits transfer event for burned gauge tokens
        ✔ emits BPT transfer event from relayer to recipient if necessary
        ✔ transfers BPT tokens to recipient
  sender = user, recipient = relayer
    when withdrawing some of the tokens
      when using immediate amounts
        ✔ pulls gauge tokens from sender if necessary
        ✔ emits BPT transfer event from gauge to relayer if necessary
        ✔ emits withdraw event
        ✔ burns gauge tokens
        ✔ emits transfer event for burned gauge tokens
        ✔ emits BPT transfer event from relayer to recipient if necessary
        ✔ transfers BPT tokens to recipient
      when using chained references
        ✔ pulls gauge tokens from sender if necessary
        ✔ emits BPT transfer event from gauge to relayer if necessary
        ✔ emits withdraw event
        ✔ burns gauge tokens
        ✔ emits transfer event for burned gauge tokens
        ✔ emits BPT transfer event from relayer to recipient if necessary
        ✔ transfers BPT tokens to recipient
    when withdrawing all the available tokens
      when using immediate amounts
        ✔ pulls gauge tokens from sender if necessary
        ✔ emits BPT transfer event from gauge to relayer if necessary
        ✔ emits withdraw event
        ✔ burns gauge tokens
        ✔ emits transfer event for burned gauge tokens
        ✔ emits BPT transfer event from relayer to recipient if necessary
        ✔ transfers BPT tokens to recipient
      when using chained references
        ✔ pulls gauge tokens from sender if necessary
        ✔ emits BPT transfer event from gauge to relayer if necessary
        ✔ emits withdraw event
        ✔ burns gauge tokens
        ✔ emits transfer event for burned gauge tokens
        ✔ emits BPT transfer event from relayer to recipient if necessary
        ✔ transfers BPT tokens to recipient
    when withdrawing 0 tokens
      when using immediate amounts
        ✔ pulls gauge tokens from sender if necessary
        ✔ emits BPT transfer event from gauge to relayer if necessary
        ✔ emits withdraw event
        ✔ burns gauge tokens
        ✔ emits transfer event for burned gauge tokens
        ✔ emits BPT transfer event from relayer to recipient if necessary
        ✔ transfers BPT tokens to recipient
      when using chained references
        ✔ pulls gauge tokens from sender if necessary
        ✔ emits BPT transfer event from gauge to relayer if necessary
        ✔ emits withdraw event
        ✔ burns gauge tokens
        ✔ emits transfer event for burned gauge tokens
        ✔ emits BPT transfer event from relayer to recipient if necessary
        ✔ transfers BPT tokens to recipient
  sender = relayer, recipient = user
    when withdrawing some of the tokens
      when using immediate amounts
        ✔ pulls gauge tokens from sender if necessary
        ✔ emits BPT transfer event from gauge to relayer if necessary
        ✔ emits withdraw event
        ✔ burns gauge tokens
        ✔ emits transfer event for burned gauge tokens
        ✔ emits BPT transfer event from relayer to recipient if necessary
        ✔ transfers BPT tokens to recipient
      when using chained references
```

```
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
      when withdrawing all the available tokens
        when using immediate amounts
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
        when using chained references
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
      when withdrawing 0 tokens
        when using immediate amounts
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
        when using chained references
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
    sender = relayer, recipient = relayer
      when withdrawing some of the tokens
        when using immediate amounts
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
        when using chained references
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
      when withdrawing all the available tokens
        when using immediate amounts
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
        when using chained references
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
      when withdrawing 0 tokens
        when using immediate amounts
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
        when using chained references
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
  gaugeMint
    when caller is approved to mint
      when not using output references
        ✔ mints BAL to sender
      when using output references
        ✔ mints BAL to sender
        ✔ stores the output in a chained reference
    when caller is not approved to mint
      ✔ reverts
  gaugeClaimRewards
    ✔ transfers rewards to sender
  gaugeCheckpoint - L1
    ✔ can call user checkpoint: false
    ✔ reverts when the user does not have a stake
    ✔ reverts when the user has not approved the relayer
    when no value is forwarded in the multicall
      ✔ checkpoints the gauges when the user has a stake
    when value is forwarded in the multicall
      ✔ checkpoints the gauges when the user has a stake
  gaugeCheckpoint - L2
    ✔ can call user checkpoint: true
    when no value is forwarded in the multicall
      ✔ checkpoints the gauges when the user has a stake
    when value is forwarded in the multicall
      ✔ checkpoints the gauges when the user has a stake
    when the user has not approved the relayer
      ✔ checkpoints the gauges when the user has a stake
Rewards only gauge
  gaugeDeposit
    when using relayer library directly
      ✔ reverts
    when caller != sender and sender != relayer
      ✔ reverts
    when sender does not have enough BPT
      ✔ reverts
    when sender has enough BPT
      sender = user, recipient = user
        when depositing some of the tokens
          when using immediate amounts
            ✔ pulls BPT tokens from sender if necessary
            ✔ approves gauge to use relayer's BPT funds
            ✔ emits BPT transfer event from relayer to gauge if necessary
            ✔ transfers BPT tokens to gauge
            ✔ emits deposit event
            ✔ mints gauge tokens to recipient
            ✔ emits transfer event for minted gauge tokens
          when using chained references
            ✔ pulls BPT tokens from sender if necessary
            ✔ approves gauge to use relayer's BPT funds
            ✔ emits BPT transfer event from relayer to gauge if necessary
            ✔ transfers BPT tokens to gauge
        ✔ emits deposit event
        ✔ mints gauge tokens to recipient
        ✔ emits transfer event for minted gauge tokens
    when depositing all of the available tokens
      when using immediate amounts
        ✔ pulls BPT tokens from sender if necessary
        ✔ approves gauge to use relayer's BPT funds
        ✔ emits BPT transfer event from relayer to gauge if necessary
        ✔ transfers BPT tokens to gauge
        ✔ emits deposit event
        ✔ mints gauge tokens to recipient
        ✔ emits transfer event for minted gauge tokens
      when using chained references
        ✔ pulls BPT tokens from sender if necessary
        ✔ approves gauge to use relayer's BPT funds
        ✔ emits BPT transfer event from relayer to gauge if necessary
        ✔ transfers BPT tokens to gauge
        ✔ emits deposit event
        ✔ mints gauge tokens to recipient
        ✔ emits transfer event for minted gauge tokens
    when depositing 0 tokens
      when using immediate amounts
        ✔ pulls BPT tokens from sender if necessary
        ✔ approves gauge to use relayer's BPT funds
        ✔ emits BPT transfer event from relayer to gauge if necessary
        ✔ transfers BPT tokens to gauge
        ✔ emits deposit event
        ✔ mints gauge tokens to recipient
        ✔ emits transfer event for minted gauge tokens
      when using chained references
        ✔ pulls BPT tokens from sender if necessary
        ✔ approves gauge to use relayer's BPT funds
        ✔ emits BPT transfer event from relayer to gauge if necessary
        ✔ transfers BPT tokens to gauge
        ✔ emits deposit event
        ✔ mints gauge tokens to recipient
        ✔ emits transfer event for minted gauge tokens
  sender = user, recipient = relayer
    when depositing some of the tokens
      when using immediate amounts
        ✔ pulls BPT tokens from sender if necessary
        ✔ approves gauge to use relayer's BPT funds
        ✔ emits BPT transfer event from relayer to gauge if necessary
        ✔ transfers BPT tokens to gauge
        ✔ emits deposit event
        ✔ mints gauge tokens to recipient
        ✔ emits transfer event for minted gauge tokens
      when using chained references
        ✔ pulls BPT tokens from sender if necessary
        ✔ approves gauge to use relayer's BPT funds
        ✔ emits BPT transfer event from relayer to gauge if necessary
        ✔ transfers BPT tokens to gauge
        ✔ emits deposit event
        ✔ mints gauge tokens to recipient
        ✔ emits transfer event for minted gauge tokens
    when depositing all of the available tokens
      when using immediate amounts
        ✔ pulls BPT tokens from sender if necessary
        ✔ approves gauge to use relayer's BPT funds
        ✔ emits BPT transfer event from relayer to gauge if necessary
        ✔ transfers BPT tokens to gauge
        ✔ emits deposit event
        ✔ mints gauge tokens to recipient
        ✔ emits transfer event for minted gauge tokens
      when using chained references
        ✔ pulls BPT tokens from sender if necessary
        ✔ approves gauge to use relayer's BPT funds
        ✔ emits BPT transfer event from relayer to gauge if necessary
        ✔ transfers BPT tokens to gauge
        ✔ emits deposit event
        ✔ mints gauge tokens to recipient
        ✔ emits transfer event for minted gauge tokens
        when depositing 0 tokens
          when using immediate amounts
            ✔ pulls BPT tokens from sender if necessary
            ✔ approves gauge to use relayer's BPT funds
            ✔ emits BPT transfer event from relayer to gauge if necessary
            ✔ transfers BPT tokens to gauge
            ✔ emits deposit event
            ✔ mints gauge tokens to recipient
            ✔ emits transfer event for minted gauge tokens
          when using chained references
            ✔ pulls BPT tokens from sender if necessary
            ✔ approves gauge to use relayer's BPT funds
            ✔ emits BPT transfer event from relayer to gauge if necessary
            ✔ transfers BPT tokens to gauge
            ✔ emits deposit event
            ✔ mints gauge tokens to recipient
            ✔ emits transfer event for minted gauge tokens
      sender = relayer, recipient = user
        when depositing some of the tokens
          when using immediate amounts
            ✔ pulls BPT tokens from sender if necessary
            ✔ approves gauge to use relayer's BPT funds
            ✔ emits BPT transfer event from relayer to gauge if necessary
            ✔ transfers BPT tokens to gauge
            ✔ emits deposit event
            ✔ mints gauge tokens to recipient
            ✔ emits transfer event for minted gauge tokens
          when using chained references
            ✔ pulls BPT tokens from sender if necessary
            ✔ approves gauge to use relayer's BPT funds
            ✔ emits BPT transfer event from relayer to gauge if necessary
            ✔ transfers BPT tokens to gauge
            ✔ emits deposit event
            ✔ mints gauge tokens to recipient
            ✔ emits transfer event for minted gauge tokens
        when depositing all of the available tokens
          when using immediate amounts
            ✔ pulls BPT tokens from sender if necessary
            ✔ approves gauge to use relayer's BPT funds
            ✔ emits BPT transfer event from relayer to gauge if necessary
            ✔ transfers BPT tokens to gauge
            ✔ emits deposit event
            ✔ mints gauge tokens to recipient
            ✔ emits transfer event for minted gauge tokens
          when using chained references
            ✔ pulls BPT tokens from sender if necessary
            ✔ approves gauge to use relayer's BPT funds
            ✔ emits BPT transfer event from relayer to gauge if necessary
            ✔ transfers BPT tokens to gauge
            ✔ emits deposit event
            ✔ mints gauge tokens to recipient
            ✔ emits transfer event for minted gauge tokens
        when depositing 0 tokens
          when using immediate amounts
            ✔ pulls BPT tokens from sender if necessary
            ✔ approves gauge to use relayer's BPT funds
            ✔ emits BPT transfer event from relayer to gauge if necessary
            ✔ transfers BPT tokens to gauge
            ✔ emits deposit event
            ✔ mints gauge tokens to recipient
            ✔ emits transfer event for minted gauge tokens
          when using chained references
            ✔ pulls BPT tokens from sender if necessary
            ✔ approves gauge to use relayer's BPT funds
            ✔ emits BPT transfer event from relayer to gauge if necessary
            ✔ transfers BPT tokens to gauge
            ✔ emits deposit event
            ✔ mints gauge tokens to recipient
            ✔ emits transfer event for minted gauge tokens
      sender = relayer, recipient = relayer
        when depositing some of the tokens
          when using immediate amounts
          ✔ pulls BPT tokens from sender if necessary
          ✔ approves gauge to use relayer's BPT funds
          ✔ emits BPT transfer event from relayer to gauge if necessary
          ✔ transfers BPT tokens to gauge
          ✔ emits deposit event
          ✔ mints gauge tokens to recipient
          ✔ emits transfer event for minted gauge tokens
        when using chained references
          ✔ pulls BPT tokens from sender if necessary
          ✔ approves gauge to use relayer's BPT funds
          ✔ emits BPT transfer event from relayer to gauge if necessary
          ✔ transfers BPT tokens to gauge
          ✔ emits deposit event
          ✔ mints gauge tokens to recipient
          ✔ emits transfer event for minted gauge tokens
      when depositing all of the available tokens
        when using immediate amounts
          ✔ pulls BPT tokens from sender if necessary
          ✔ approves gauge to use relayer's BPT funds
          ✔ emits BPT transfer event from relayer to gauge if necessary
          ✔ transfers BPT tokens to gauge
          ✔ emits deposit event
          ✔ mints gauge tokens to recipient
          ✔ emits transfer event for minted gauge tokens
        when using chained references
          ✔ pulls BPT tokens from sender if necessary
          ✔ approves gauge to use relayer's BPT funds
          ✔ emits BPT transfer event from relayer to gauge if necessary
          ✔ transfers BPT tokens to gauge
          ✔ emits deposit event
          ✔ mints gauge tokens to recipient
          ✔ emits transfer event for minted gauge tokens
      when depositing 0 tokens
        when using immediate amounts
          ✔ pulls BPT tokens from sender if necessary
          ✔ approves gauge to use relayer's BPT funds
          ✔ emits BPT transfer event from relayer to gauge if necessary
          ✔ transfers BPT tokens to gauge
          ✔ emits deposit event
          ✔ mints gauge tokens to recipient
          ✔ emits transfer event for minted gauge tokens
        when using chained references
          ✔ pulls BPT tokens from sender if necessary
          ✔ approves gauge to use relayer's BPT funds
          ✔ emits BPT transfer event from relayer to gauge if necessary
          ✔ transfers BPT tokens to gauge
          ✔ emits deposit event
          ✔ mints gauge tokens to recipient
          ✔ emits transfer event for minted gauge tokens
  gaugeWithdraw
    when using relayer library directly
      ✔ reverts
    when caller != sender and sender != relayer
      ✔ reverts
    when sender does not have enough gauge tokens
      ✔ reverts
    when sender has enough gauge tokens
      sender = user, recipient = user
        when withdrawing some of the tokens
          when using immediate amounts
            ✔ pulls gauge tokens from sender if necessary
            ✔ emits BPT transfer event from gauge to relayer if necessary
            ✔ emits withdraw event
            ✔ burns gauge tokens
            ✔ emits transfer event for burned gauge tokens
            ✔ emits BPT transfer event from relayer to recipient if necessary
            ✔ transfers BPT tokens to recipient
          when using chained references
            ✔ pulls gauge tokens from sender if necessary
            ✔ emits BPT transfer event from gauge to relayer if necessary
            ✔ emits withdraw event
            ✔ burns gauge tokens
        ✔ emits transfer event for burned gauge tokens
        ✔ emits BPT transfer event from relayer to recipient if necessary
        ✔ transfers BPT tokens to recipient
    when withdrawing all the available tokens
      when using immediate amounts
        ✔ pulls gauge tokens from sender if necessary
        ✔ emits BPT transfer event from gauge to relayer if necessary
        ✔ emits withdraw event
        ✔ burns gauge tokens
        ✔ emits transfer event for burned gauge tokens
        ✔ emits BPT transfer event from relayer to recipient if necessary
        ✔ transfers BPT tokens to recipient
      when using chained references
        ✔ pulls gauge tokens from sender if necessary
        ✔ emits BPT transfer event from gauge to relayer if necessary
        ✔ emits withdraw event
        ✔ burns gauge tokens
        ✔ emits transfer event for burned gauge tokens
        ✔ emits BPT transfer event from relayer to recipient if necessary
        ✔ transfers BPT tokens to recipient
    when withdrawing 0 tokens
      when using immediate amounts
        ✔ pulls gauge tokens from sender if necessary
        ✔ emits BPT transfer event from gauge to relayer if necessary
        ✔ emits withdraw event
        ✔ burns gauge tokens
        ✔ emits transfer event for burned gauge tokens
        ✔ emits BPT transfer event from relayer to recipient if necessary
        ✔ transfers BPT tokens to recipient
      when using chained references
        ✔ pulls gauge tokens from sender if necessary
        ✔ emits BPT transfer event from gauge to relayer if necessary
        ✔ emits withdraw event
        ✔ burns gauge tokens
        ✔ emits transfer event for burned gauge tokens
        ✔ emits BPT transfer event from relayer to recipient if necessary
        ✔ transfers BPT tokens to recipient
  sender = user, recipient = relayer
    when withdrawing some of the tokens
      when using immediate amounts
        ✔ pulls gauge tokens from sender if necessary
        ✔ emits BPT transfer event from gauge to relayer if necessary
        ✔ emits withdraw event
        ✔ burns gauge tokens
        ✔ emits transfer event for burned gauge tokens
        ✔ emits BPT transfer event from relayer to recipient if necessary
        ✔ transfers BPT tokens to recipient
      when using chained references
        ✔ pulls gauge tokens from sender if necessary
        ✔ emits BPT transfer event from gauge to relayer if necessary
        ✔ emits withdraw event
        ✔ burns gauge tokens
        ✔ emits transfer event for burned gauge tokens
        ✔ emits BPT transfer event from relayer to recipient if necessary
        ✔ transfers BPT tokens to recipient
    when withdrawing all the available tokens
      when using immediate amounts
        ✔ pulls gauge tokens from sender if necessary
        ✔ emits BPT transfer event from gauge to relayer if necessary
        ✔ emits withdraw event
        ✔ burns gauge tokens
        ✔ emits transfer event for burned gauge tokens
        ✔ emits BPT transfer event from relayer to recipient if necessary
        ✔ transfers BPT tokens to recipient
      when using chained references
        ✔ pulls gauge tokens from sender if necessary
        ✔ emits BPT transfer event from gauge to relayer if necessary
        ✔ emits withdraw event
        ✔ burns gauge tokens
        ✔ emits transfer event for burned gauge tokens
        ✔ emits BPT transfer event from relayer to recipient if necessary
        ✔ transfers BPT tokens to recipient
      when withdrawing 0 tokens
        when using immediate amounts
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
        when using chained references
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
    sender = relayer, recipient = user
      when withdrawing some of the tokens
        when using immediate amounts
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
        when using chained references
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
      when withdrawing all the available tokens
        when using immediate amounts
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
        when using chained references
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
      when withdrawing 0 tokens
        when using immediate amounts
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
        when using chained references
          ✔ pulls gauge tokens from sender if necessary
          ✔ emits BPT transfer event from gauge to relayer if necessary
          ✔ emits withdraw event
          ✔ burns gauge tokens
          ✔ emits transfer event for burned gauge tokens
          ✔ emits BPT transfer event from relayer to recipient if necessary
          ✔ transfers BPT tokens to recipient
    sender = relayer, recipient = relayer
      when withdrawing some of the tokens
        when using immediate amounts
```

```
            ✔ pulls gauge tokens from sender if necessary
            ✔ emits BPT transfer event from gauge to relayer if necessary
            ✔ emits withdraw event
            ✔ burns gauge tokens
            ✔ emits transfer event for burned gauge tokens
            ✔ emits BPT transfer event from relayer to recipient if necessary
            ✔ transfers BPT tokens to recipient
          when using chained references
            ✔ pulls gauge tokens from sender if necessary
            ✔ emits BPT transfer event from gauge to relayer if necessary
            ✔ emits withdraw event
            ✔ burns gauge tokens
            ✔ emits transfer event for burned gauge tokens
            ✔ emits BPT transfer event from relayer to recipient if necessary
            ✔ transfers BPT tokens to recipient
        when withdrawing all the available tokens
          when using immediate amounts
            ✔ pulls gauge tokens from sender if necessary
            ✔ emits BPT transfer event from gauge to relayer if necessary
            ✔ emits withdraw event
            ✔ burns gauge tokens
            ✔ emits transfer event for burned gauge tokens
            ✔ emits BPT transfer event from relayer to recipient if necessary
            ✔ transfers BPT tokens to recipient
          when using chained references
            ✔ pulls gauge tokens from sender if necessary
            ✔ emits BPT transfer event from gauge to relayer if necessary
            ✔ emits withdraw event
            ✔ burns gauge tokens
            ✔ emits transfer event for burned gauge tokens
            ✔ emits BPT transfer event from relayer to recipient if necessary
            ✔ transfers BPT tokens to recipient
        when withdrawing 0 tokens
          when using immediate amounts
            ✔ pulls gauge tokens from sender if necessary
            ✔ emits BPT transfer event from gauge to relayer if necessary
            ✔ emits withdraw event
            ✔ burns gauge tokens
            ✔ emits transfer event for burned gauge tokens
            ✔ emits BPT transfer event from relayer to recipient if necessary
            ✔ transfers BPT tokens to recipient
          when using chained references
            ✔ pulls gauge tokens from sender if necessary
            ✔ emits BPT transfer event from gauge to relayer if necessary
            ✔ emits withdraw event
            ✔ burns gauge tokens
            ✔ emits transfer event for burned gauge tokens
            ✔ emits BPT transfer event from relayer to recipient if necessary
            ✔ transfers BPT tokens to recipient
    gaugeClaimRewards
      ✔ first transfers rewards to gauge
      ✔ then transfers rewards to sender

GearboxWrapping
  primitives
    wrapGearbox
      sender = senderUser, recipient = relayer
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
      sender = senderUser, recipient = senderUser
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
      sender = relayer, recipient = relayer
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
      sender = relayer, recipient = senderUser
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
    unwrapGearbox
      sender = senderUser, recipient = relayer
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
      sender = senderUser, recipient = senderUser
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
      sender = relayer, recipient = relayer
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
      sender = relayer, recipient = senderUser
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
  complex actions
    swap
      swap using DAI as an input
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
      swap using DAI as an output
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
    batchSwap
      swap using DAI as an input
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
      swap using DAI as an output
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
    joinPool
      ✔ joins the pool
      ✔ does not take dDAI from the user
      ✔ does not leave dust on the relayer
    exitPool
      ✔ exits the pool
      ✔ BPT burned from the sender user
      ✔ DAI transfered to recipient user
      ✔ does not leave dust on the relayer

LidoWrapping
  primitives
    wrapStETH
      sender = senderUser, recipient = relayer
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
      sender = senderUser, recipient = senderUser
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
      sender = relayer, recipient = relayer
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
      sender = relayer, recipient = senderUser
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
    unwrapWstETH
      sender = senderUser, recipient = relayer
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
      sender = senderUser, recipient = senderUser
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
      sender = relayer, recipient = relayer
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
      sender = relayer, recipient = senderUser
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
    stakeETH
      recipient = senderUser
        ✔ stakes with immediate amounts
        ✔ returns excess ETH
        ✔ stores stake output as chained reference
        ✔ stakes with chained references
      recipient = relayer
        ✔ stakes with immediate amounts
        ✔ returns excess ETH
        ✔ stores stake output as chained reference
        ✔ stakes with chained references
    stakeETHAndWrap
      recipient = senderUser
        ✔ stakes with immediate amounts
        ✔ stores stake output as chained reference
        ✔ stakes with chained references
      recipient = relayer
        ✔ stakes with immediate amounts
        ✔ stores stake output as chained reference
        ✔ stakes with chained references
  complex actions
    swap
      swap using stETH as an input
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
      swap using stETH as an output
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
    batchSwap
      swap using stETH as an input
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
      swap using stETH as an output
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
    joinPool
      ✔ joins the pool
      ✔ does not take wstETH from the user
      ✔ does not leave dust on the relayer

PoolCreationHelper
  check set state
    ✔ should set the state correctly
  createAndJoinWeightedPool
    ✔ creates and joins a weighted pool
    ✔ create and join weighted pool with same token more than once
    ✔ create and join weighted `WBERA` pool with BERA (271ms)
    ✔ wbera pool creation fails if not enough BERA is sent
    ✔ should not consume BERA if wbera pool is joined with wbera (274ms)
    ✔ should not consume wbera if wbera pool is joined with BERA (279ms)
  createAndJoinStablePool
    ✔ creates and joins a composable stable pool
    ✔ create and join WBERA pool with BERA (341ms)
    ✔ wbera pool creation fails if not enough BERA is sent
    ✔ should not consume BERA if wbera pool is joined with wbera (291ms)
    ✔ should not consume wbera if wbera pool is joined with BERA (281ms)

PoolRecoveryHelper
  constructor
    ✔ supports no initial factories
    ✔ stores initial factories
  factory list
    add
      ✔ reverts if the caller does not have permission
      ✔ new factories can be added
      ✔ duplicate factories are rejected
    remove
      ✔ reverts if the caller does not have permission
      ✔ existing factories can be removed
      ✔ non-existent factories are rejected
  enable recovery mode
    ✔ reverts if the pool is not from a known factory
    ✔ reverts if none of the pool's rate providers reverts
    ✔ enables recovery mode on the pool if any of the rate providers revert

ProtocolFeePercentagesProvider
  construction
    ✔ reverts if the maximum yield value is too high
    ✔ reverts if the maximum aum value is too high
    ✔ emits ProtocolFeeTypeRegistered events for custom types
    ✔ emits ProtocolFeePercentageChanged events for custom types
  with provider
    fee type configuration
      native fee types
        fee type Swap
          ✔ returns the fee type as valid
          ✔ returns the fee type name
          ✔ returns the fee type maximum value
        fee type FlashLoan
          ✔ returns the fee type as valid
          ✔ returns the fee type name
          ✔ returns the fee type maximum value
      custom fee types
        fee type Yield
          ✔ returns the fee type as valid
          ✔ returns the fee type name
          ✔ returns the fee type maximum value
          ✔ sets an initial value
        fee type AUM
          ✔ returns the fee type as valid
          ✔ returns the fee type name
          ✔ returns the fee type maximum value
          ✔ sets an initial value
      invalid fee type
        ✔ isValidFeeType returns false
        ✔ get name reverts
        ✔ get maximum reverts
    is valid fee percentage
      native fee types
        fee type Swap
          ✔ returns true if the fee is below the maximum
          ✔ returns true if the fee equals the maximum
          ✔ returns false if the fee is above the maximum
        fee type FlashLoan
          ✔ returns true if the fee is below the maximum
          ✔ returns true if the fee equals the maximum
          ✔ returns false if the fee is above the maximum
      custom fee types
        fee type Yield
          ✔ returns true if the fee is below the maximum
          ✔ returns true if the fee equals the maximum
          ✔ returns false if the fee is above the maximum
        fee type AUM
          ✔ returns true if the fee is below the maximum
          ✔ returns true if the fee equals the maximum
          ✔ returns false if the fee is above the maximum
      invalid fee type
        ✔ reverts
    set fee type value
      native fee types
        fee type Swap
          when the caller is authorized
            when the provider is authorized
              when the value is below the maximum
                ✔ sets the value
                ✔ emits a ProtocolFeePercentageChanged event
              when the value is equal to the maximum
                ✔ sets the value
                ✔ emits a ProtocolFeePercentageChanged event
              when the value is above the maximum
                ✔ reverts
            when the provider is not authorized
              ✔ reverts
          when the caller is not authorized
            ✔ reverts
        fee type FlashLoan
          when the caller is authorized
            when the provider is authorized
              when the value is below the maximum
                ✔ sets the value
                ✔ emits a ProtocolFeePercentageChanged event
              when the value is equal to the maximum
                ✔ sets the value
                ✔ emits a ProtocolFeePercentageChanged event
              when the value is above the maximum
                ✔ reverts
            when the provider is not authorized
              ✔ reverts
          when the caller is not authorized
            ✔ reverts
      custom fee types
        fee type Yield
          when the caller is authorized
            when the value is below the maximum
              ✔ sets the value
              ✔ emits a ProtocolFeePercentageChanged event
            when the value is equal to the maximum
              ✔ sets the value
              ✔ emits a ProtocolFeePercentageChanged event
            when the value is above the maximum
              ✔ reverts
          when the caller is not authorized
            ✔ reverts
        fee type AUM
          when the caller is authorized
            when the value is below the maximum
              ✔ sets the value
              ✔ emits a ProtocolFeePercentageChanged event
            when the value is equal to the maximum
              ✔ sets the value
              ✔ emits a ProtocolFeePercentageChanged event
            when the value is above the maximum
              ✔ reverts
          when the caller is not authorized
            ✔ reverts
      invalid fee type
        ✔ reverts
    native fee type out of band change
      swap fee
        ✔ the provider tracks value changes
      flash loan fee
        ✔ the provider tracks value changes
    register fee type
      when the caller is authorized
        when the fee type is already in use
          ✔ reverts
        when the maximum value is 0%
          ✔ reverts
        when the maximum value is above 100%
          ✔ reverts
        when the initial value is above the maximum value
          ✔ reverts
        when the new fee type data is valid
          ✔ returns registered data
          ✔ marks the fee type as valid
          ✔ emits a ProtocolFeeTypeRegistered event
          ✔ emits a ProtocolFeePercentageChanged event
          ✔ reverts on register attempt
          ✔ can change value after registration
      when the caller is not authorized
```

```
        ✔ reverts

  ProtocolFeeSplitter
    constructor
      ✔ sets the protocolFeesWithdrawer
      ✔ sets the treasury
    setTreasury
      ✔ changes the treasury
      ✔ emits a DAOFundsRecipientChanged event
      ✔ reverts if caller is unauthorized
    setDefaultRevenueSharePercentage
      ✔ sets default fee
      ✔ emits a DefaultRevenueSharePercentageChanged event
      ✔ reverts if caller is not authorized
    setRevenueSharingFeePercentage
      ✔ uses the default value when not set
      ✔ overrides revenue sharing percentage for a pool
      ✔ emits a PoolRevenueShareChanged event
      ✔ allows a revenue sharing percentage of zero
      ✔ reverts with invalid input
      ✔ reverts if caller is not authorized
    setPoolBeneficiary
      called by pool owner
        ✔ sets pool beneficiary
        ✔ emits a PoolBeneficiaryChanged event
      called by governance-authorized address
        ✔ sets pool beneficiary
        ✔ emits a PoolBeneficiaryChanged event
      called by other
        ✔ it reverts
    when the fee collector holds BPT
      without a beneficiary
        ✔ sends all fees to the treasury
      with a beneficiary
        ✔ emits an event with collected fees
        with no revenue share override
          ✔ should collect the default pool revenue share
        with a non-zero revenue share override
          ✔ should collect the pool revenue share
          disable revenue sharing
            ✔ emits a PoolRevenueShareCleared event
            ✔ reverts if caller is not authorized
            when revenue sharing disabled
              ✔ should now resume collecting the default revenue share
        with a zero revenue share override
          ✔ should send all funds to the treasury

  ProtocolFeesWithdrawer
    constructor
      ✔ lists the initially denylisted tokens
      ✔ reports the initial denylisted tokens as ineligible for withdrawal
    denylistToken
      ✔ adds the token to the denylist
      ✔ emits an event
      ✔ reverts if already denylisted
    allowlistToken
      ✔ removes the token from the denylist
      ✔ emits an event
      ✔ reverts if not denylisted
    withdrawCollectedFees
      when caller is authorized
        when attempting to claim allowlisted tokens
          ✔ withdraws the expected amount of tokens
        when attempting to claim denylisted tokens
          ✔ reverts
        when attempting to claim a mix of allowlisted and denylisted tokens
          ✔ reverts
        when tokens are later added from the denylist
          ✔ reverts
        when tokens are removed from the denylist
          ✔ allows withdrawing these tokens
      when caller is not authorized
```

```
        ✔ reverts
    distributeAndWithdrawCollectedFees
      when caller is authorized
        ✔ withdraws allowlisted tokens to POL fee collector
        ✔ withdraw allowlist tokens with `polFeeCollectorPercentage` as 50%
        ✔ reverts when attempting to withdraw denylisted tokens
        when attempting to claim a mix of allowlisted and denylisted tokens
          ✔ reverts
        when tokens are later added from the denylist
          ✔ reverts
        when tokens are removed from the denylist
          ✔ allows withdrawing these tokens
      when caller is not authorized
        ✔ reverts
    set PolFeeCollector
      ✔ caller is allowed to set the polFeeCollector
      ✔ reverts if zero address
      ✔ reverts if sender not allowed
    set feeReceiver
      ✔ caller is allowed to set the feeReceiver
      ✔ reverts if zero address
      ✔ reverts if sender not allowed
    set polFeeCollectorPercentage
      ✔ caller is allowed to set the polFeeCollectorPercentage
      ✔ reverts if percentage is greater than 100%
      ✔ reverts if sender not allowed

  ProtocolIdRegistry
    Constructor
      ✔ events are emitted for protocols initialized in the constructor
      Aave v1 protocol is registered with protocol id 0
        ✔ Protocol Id is valid
        ✔ Protocol name is correct
    Registration
      authorized user
        ✔ event emitted
        ✔ new ID is valid
        ✔ name matches ID
        ✔ reverts when registering existing ID
      non-authorized user
        ✔ registration gets reverted
    Unregistered queries
      ✔ searching for name in non-existent protocol ID
      ✔ check non-valid ID
    rename protocol IDs
      when the user is authorized to rename
        ✔ emits an event
        ✔ renames existing protocol ID
        ✔ reverts renaming non-existing protocol ID
      when the user is not authorized to rename
        ✔ reverts

  ReaperWrapping
    wrapping
      ✔ should deposit underlying tokens into a reaper vault on wrap
      ✔ should leave yv tokens on the relayer, when the recipient of the wrap is the relayer
      ✔ stores wrap output as chained reference
      ✔ wraps with chained references
    unwrapping
      ✔ should withdraw underlying tokens from a reaper vault on unwrap
      ✔ should leave tokens on the relayer, when the recipient of the unwrap is the relayer
      ✔ stores unwrap output as chained reference

  SiloWrapping
    primitives
      wrap DAI
        sender = senderUser, recipient = relayer
          ✔ wraps with immediate amounts
          ✔ stores wrap output as chained reference
          ✔ wraps with chained references
        sender = senderUser, recipient = senderUser
          ✔ wraps with immediate amounts
          ✔ stores wrap output as chained reference
          ✔ wraps with chained references
        sender = relayer, recipient = relayer
          ✔ wraps with immediate amounts
          ✔ stores wrap output as chained reference
          ✔ wraps with chained references
        sender = relayer, recipient = senderUser
          ✔ wraps with immediate amounts
          ✔ stores wrap output as chained reference
          ✔ wraps with chained references
      unwrap sDAI
        sender = senderUser, recipient = relayer
          ✔ unwraps with immediate amounts
          ✔ stores unwrap output as chained reference
          ✔ unwraps with chained references
        sender = senderUser, recipient = senderUser
          ✔ unwraps with immediate amounts
          ✔ stores unwrap output as chained reference
          ✔ unwraps with chained references
        sender = relayer, recipient = relayer
          ✔ unwraps with immediate amounts
          ✔ stores unwrap output as chained reference
          ✔ unwraps with chained references
        sender = relayer, recipient = senderUser
          ✔ unwraps with immediate amounts
          ✔ stores unwrap output as chained reference
          ✔ unwraps with chained references
    complex actions
      swap
        swap using DAI as an input
          ✔ performs the given swap
          ✔ does not leave dust on the relayer
        swap using DAI as an output
          ✔ performs the given swap
          ✔ does not leave dust on the relayer
      batchSwap
        swap using DAI as an input
          ✔ performs the given swap
          ✔ does not leave dust on the relayer
        swap using DAI as an output
          ✔ performs the given swap
          ✔ does not leave dust on the relayer
      joinPool
        ✔ joins the pool
        ✔ does not take sDAI from the user
        ✔ does not leave dust on the relayer
      exitPool
        ✔ exits the pool
        ✔ BPT burned from the sender user
        ✔ DAI transfered to recipient user
        ✔ does not leave dust on the relayer

  SpotPriceOracle
    #getAssetPrice(address)
      ✔ price of USDC in PYUSD matches price derived from math
      ✔ price of PYUSD in USDC matches price derived from math
      reverts when...
        ✔ asset address is neither USDC nor PYUSD
    #getPriceNoOlderThan(address, uint256)
      ✔ returns the same value as getPrice
      reverts when...
        ✔ age is not 0
    #getPriceUnsafe(address)
      ✔ returns the same value as getPrice
    #priceAvailable(address)
      ✔ returns true if asset is USDC or PYUSD
      ✔ returns false if asset is neither USDC nor PYUSD

  TetuWrapping
    primitives
      wrapTetu
        sender = senderUser, recipient = relayer
```

```
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
      sender = senderUser, recipient = senderUser
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
      sender = relayer, recipient = relayer
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
      sender = relayer, recipient = senderUser
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
    unwrapTetu
      sender = senderUser, recipient = relayer
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
      sender = senderUser, recipient = senderUser
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
      sender = relayer, recipient = relayer
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
      sender = relayer, recipient = senderUser
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
  complex actions
    swap
      swap using DAI as an input
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
      swap using DAI as an output
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
    batchSwap
      swap using DAI as an input
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
      swap using DAI as an output
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
    joinPool
      ✔ joins the pool
      ✔ does not take xDAI from the user
      ✔ does not leave dust on the relayer
    exitPool
      ✔ exits the pool
      ✔ BPT burned from the sender user
      ✔ DAI transfered to recipient user
      ✔ does not leave dust on the relayer

UnbuttonWrapping
  primitives
    wrap AMPL
      sender = senderUser, recipient = relayer
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
      sender = senderUser, recipient = senderUser
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
      sender = relayer, recipient = relayer
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
      sender = relayer, recipient = senderUser
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
    unwrap WAMPL
      sender = senderUser, recipient = relayer
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
      sender = senderUser, recipient = senderUser
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
      sender = relayer, recipient = relayer
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
      sender = relayer, recipient = senderUser
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
  complex actions
    swap
      swap using ampl as an input
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
      swap using ampl as an output
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
    batchSwap
      swap using ampl as an input
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
      swap using ampl as an output
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
    joinPool
      ✔ joins the pool
      ✔ does not take wampl from the user
      ✔ does not leave dust on the relayer

VaultActions
  simple swap
    when caller is not authorized
      ✔ reverts
    when caller is authorized
      sender = user
        ✔ swaps with immediate amounts
        ✔ stores swap output as chained reference
        ✔ swaps with chained references
        ✔ is chainable via multicall
      sender = relayer
        ✔ swaps with immediate amounts
        ✔ stores swap output as chained reference
        ✔ swaps with chained references
        ✔ is chainable via multicall
  batch swap
    when caller is not authorized
      ✔ reverts
    when caller is authorized
      sender = user
        ✔ swaps with immediate amounts
        ✔ stores absolute vault deltas as chained reference
        ✔ swaps with chained references
        ✔ is chainable via multicall
      sender = relayer
        ✔ swaps with immediate amounts
        ✔ stores absolute vault deltas as chained reference
        ✔ swaps with chained references
        ✔ is chainable via multicall
  join pool
    when caller is not authorized
      ✔ reverts
    when caller is authorized
      weighted pool
        sender = user
          exact tokens in for bpt out
            ✔ joins with immediate amounts
            ✔ stores BPT amount out as chained reference
            ✔ joins with exact amounts in chained references
            ✔ is chainable with swaps via multicall
          token in for exact bpt out
            ✔ joins with immediate amounts
          all tokens in for exact bpt out
            ✔ joins with immediate amounts
        sender = relayer
          exact tokens in for bpt out
            ✔ joins with immediate amounts
            ✔ stores BPT amount out as chained reference
            ✔ joins with exact amounts in chained references
            ✔ is chainable with swaps via multicall
          token in for exact bpt out
            ✔ joins with immediate amounts
          all tokens in for exact bpt out
            ✔ joins with immediate amounts
  exit pool
    when caller is not authorized
      ✔ reverts
    when caller is authorized
      weighted pool
        sender = user
          exit to external balance
            exact bpt in for tokens
              ✔ exits with immediate amounts
              ✔ stores token amount out as chained reference
              ✔ exits with exact bpt in chained reference
              ✔ is chainable with swaps via multicall
            exact bpt in for one token
              ✔ exits with immediate amounts
              ✔ stores token amount out as chained reference
              ✔ exits with exact bpt in chained reference
              ✔ is chainable with swaps via multicall
            bpt in for exact tokens out
              ✔ exits with immediate amounts
          exit to internal balance
            exact bpt in for tokens
              ✔ exits with immediate amounts
              ✔ stores token amount out as chained reference
              ✔ exits with exact bpt in chained reference
              ✔ is chainable with swaps via multicall
            exact bpt in for one token
              ✔ exits with immediate amounts
              ✔ stores token amount out as chained reference
              ✔ exits with exact bpt in chained reference
              ✔ is chainable with swaps via multicall
            bpt in for exact tokens out
              ✔ exits with immediate amounts
        sender = relayer
          exit to external balance
            exact bpt in for tokens
              ✔ exits with immediate amounts
              ✔ stores token amount out as chained reference
              ✔ exits with exact bpt in chained reference
              ✔ is chainable with swaps via multicall
            exact bpt in for one token
              ✔ exits with immediate amounts
              ✔ stores token amount out as chained reference
              ✔ exits with exact bpt in chained reference
              ✔ is chainable with swaps via multicall
            bpt in for exact tokens out
              ✔ exits with immediate amounts
          exit to internal balance
            exact bpt in for tokens
              ✔ exits with immediate amounts
              ✔ stores token amount out as chained reference
              ✔ exits with exact bpt in chained reference
              ✔ is chainable with swaps via multicall
            exact bpt in for one token
              ✔ exits with immediate amounts
              ✔ stores token amount out as chained reference
              ✔ exits with exact bpt in chained reference
              ✔ is chainable with swaps via multicall
            bpt in for exact tokens out
              ✔ exits with immediate amounts
    exit pool in recovery mode
      when caller is not authorized
        ✔ reverts
      when caller is authorized
        weighted pool
          sender = user
            exit to external balance
              exact bpt in for all tokens
                ✔ exits with immediate amounts
                ✔ stores token amount out as chained reference
                ✔ exits with exact bpt in chained reference
                ✔ is chainable with swaps via multicall
            exit to internal balance
              exact bpt in for all tokens
                ✔ exits with immediate amounts
                ✔ stores token amount out as chained reference
                ✔ exits with exact bpt in chained reference
                ✔ is chainable with swaps via multicall
          sender = relayer
            exit to external balance
              exact bpt in for all tokens
                ✔ exits with immediate amounts
                ✔ stores token amount out as chained reference
                ✔ exits with exact bpt in chained reference
                ✔ is chainable with swaps via multicall
            exit to internal balance
              exact bpt in for all tokens
                ✔ exits with immediate amounts
                ✔ stores token amount out as chained reference
                ✔ exits with exact bpt in chained reference
                ✔ is chainable with swaps via multicall
    user balance ops
      when caller is not authorized
        ✔ reverts
      when caller is authorized
        sender = user
          ✔ sends immediate amounts
          ✔ stores vault deltas as chained references
          ✔ emits internal balance events
          ✔ uses chained references
          ✔ is chainable via multicall
          ✔ allows emergency exit
        sender = relayer
          ✔ sends immediate amounts
          ✔ stores vault deltas as chained references
          ✔ emits internal balance events
          ✔ uses chained references
          ✔ is chainable via multicall
          ✔ allows emergency exit

  Vault Actions - Stable Pools
    join pool
      when caller is not authorized
        ✔ reverts
      when caller is authorized
        sender = user
          exact tokens in for bpt out
            ✔ joins with immediate amounts
            ✔ stores BPT amount out as chained reference
            ✔ joins with exact amounts in chained references
            ✔ is chainable with swaps via multicall
          token in for exact bpt out
        ✔ joins with immediate amounts
      all tokens in for exact bpt out
        ✔ joins with immediate amounts
    sender = relayer
      exact tokens in for bpt out
        ✔ joins with immediate amounts
        ✔ stores BPT amount out as chained reference
        ✔ joins with exact amounts in chained references
        ✔ is chainable with swaps via multicall
      token in for exact bpt out
        ✔ joins with immediate amounts
      all tokens in for exact bpt out
        ✔ joins with immediate amounts
exit pool
  when caller is not authorized
    ✔ reverts
  when caller is authorized
    sender = user
      exit to external balance
        exact bpt in for tokens
          ✔ exits with immediate amounts
          ✔ stores token amount out as chained reference
          ✔ exits with exact bpt in chained reference
          ✔ is chainable with swaps via multicall
        exact bpt in for one token
          ✔ exits with immediate amounts
          ✔ stores token amount out as chained reference
          ✔ exits with exact bpt in chained reference
          ✔ is chainable with swaps via multicall
        bpt in for exact tokens out
          ✔ exits with immediate amounts
      exit to internal balance
        exact bpt in for tokens
          ✔ exits with immediate amounts
          ✔ stores token amount out as chained reference
          ✔ exits with exact bpt in chained reference
          ✔ is chainable with swaps via multicall
        exact bpt in for one token
          ✔ exits with immediate amounts
          ✔ stores token amount out as chained reference
          ✔ exits with exact bpt in chained reference
          ✔ is chainable with swaps via multicall
        bpt in for exact tokens out
          ✔ exits with immediate amounts
    sender = relayer
      exit to external balance
        exact bpt in for tokens
          ✔ exits with immediate amounts
          ✔ stores token amount out as chained reference
          ✔ exits with exact bpt in chained reference
          ✔ is chainable with swaps via multicall
        exact bpt in for one token
          ✔ exits with immediate amounts
          ✔ stores token amount out as chained reference
          ✔ exits with exact bpt in chained reference
          ✔ is chainable with swaps via multicall
        bpt in for exact tokens out
          ✔ exits with immediate amounts
      exit to internal balance
        exact bpt in for tokens
          ✔ exits with immediate amounts
          ✔ stores token amount out as chained reference
          ✔ exits with exact bpt in chained reference
          ✔ is chainable with swaps via multicall
        exact bpt in for one token
          ✔ exits with immediate amounts
          ✔ stores token amount out as chained reference
          ✔ exits with exact bpt in chained reference
          ✔ is chainable with swaps via multicall
        bpt in for exact tokens out
          ✔ exits with immediate amounts
exit pool in recovery mode
```

```
    when caller is not authorized
      ✔ reverts
    when caller is authorized
      sender = user
        exit to external balance
          exact bpt in for all tokens
            ✔ exits with immediate amounts
            ✔ stores token amount out as chained reference
            ✔ exits with exact bpt in chained reference
            ✔ is chainable with swaps via multicall
        exit to internal balance
          exact bpt in for all tokens
            ✔ exits with immediate amounts
            ✔ stores token amount out as chained reference
            ✔ exits with exact bpt in chained reference
            ✔ is chainable with swaps via multicall
      sender = relayer
        exit to external balance
          exact bpt in for all tokens
            ✔ exits with immediate amounts
            ✔ stores token amount out as chained reference
            ✔ exits with exact bpt in chained reference
            ✔ is chainable with swaps via multicall
        exit to internal balance
          exact bpt in for all tokens
            ✔ exits with immediate amounts
            ✔ stores token amount out as chained reference
            ✔ exits with exact bpt in chained reference
            ✔ is chainable with swaps via multicall
  unhandled pool types
    on joins
      ✔ does not support invalid pool types on joins
    on exits
      ✔ does not support invalid pool types on exits

VaultQueryActions
  simple swap
    when caller is not authorized
      ✔ reverts
    when caller is authorized
      sender = user
        simple swap
          ✔ stores swap output as chained reference
          ✔ returns the swap output directly
      sender = relayer
        simple swap
          ✔ stores swap output as chained reference
          ✔ returns the swap output directly
  batch swap
    when caller is not authorized
      ✔ reverts
    when caller is authorized
      sender = user
        batch swap
          ✔ stores batch swap output as chained reference
          ✔ stores batch swap output directly
      sender = relayer
        batch swap
          ✔ stores batch swap output as chained reference
          ✔ stores batch swap output directly
  join
    when caller is not authorized
      ✔ reverts
    when caller is authorized
      sender = user
        ✔ stores join result as chained reference
      sender = relayer
        ✔ stores join result as chained reference
  exit
    when caller is not authorized
      ✔ reverts
    when caller is authorized
```

```
      sender = user
        ✔ stores exit result as chained reference
      sender = relayer
        ✔ stores exit result as chained reference
  user balance ops
    ✔ does not allow calls to manageUserBalance

YearnWrapping
  primitives
    wrapYearn
      sender = senderUser, recipient = relayer
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
      sender = senderUser, recipient = senderUser
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
      sender = relayer, recipient = relayer
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
      sender = relayer, recipient = senderUser
        ✔ wraps with immediate amounts
        ✔ stores wrap output as chained reference
        ✔ wraps with chained references
    unwrapYearn
      sender = senderUser, recipient = relayer
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
      sender = senderUser, recipient = senderUser
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
      sender = relayer, recipient = relayer
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
      sender = relayer, recipient = senderUser
        ✔ unwraps with immediate amounts
        ✔ stores unwrap output as chained reference
        ✔ unwraps with chained references
  complex actions
    swap
      swap using DAI as an input
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
      swap using DAI as an output
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
    batchSwap
      swap using DAI as an input
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
      swap using DAI as an output
        ✔ performs the given swap
        ✔ does not leave dust on the relayer
    joinPool
      ✔ joins the pool
      ✔ does not take yvDAI from the user
      ✔ does not leave dust on the relayer
    exitPool
      ✔ exits the pool
      ✔ BPT burned from the sender user
      ✔ DAI transfered to recipient user
      ✔ does not leave dust on the relayer


1493 passing (1m)
```

Code Coverage

For the `@balancer-labs/v2-standalone-utils` package, test coverage is very close to 100% for the modified contracts `ProtocolFeePercentagesProvider.sol` and `ProtocolFeesWithdrawer.sol`, as well as the new contracts `PoolCreationHelper.sol` and `SpotPriceOracle.sol`.

```
-------------------------------------|----------|----------|----------|----------|----------------|
File                                 |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
-------------------------------------|----------|----------|----------|----------|----------------|
 contracts/                          |     61.8 |     66.5 |    76.32 |    61.23 |                |
  BALTokenHolder.sol                 |      100 |      100 |      100 |      100 |                |
  BALTokenHolderFactory.sol          |      100 |      100 |      100 |      100 |                |
  BalancerPoolDataQueries.sol        |        0 |        0 |        0 |        0 |... 566,567,571 |
  BalancerQueries.sol                |       95 |       50 |      100 |     96.3 |             78 |
  BatchRelayerLibrary.sol            |      100 |      100 |      100 |      100 |                |
  BatchRelayerQueryLibrary.sol       |      100 |      100 |      100 |      100 |                |
  PoolCreationHelper.sol             |      100 |      100 |      100 |      100 |                |
  PoolRecoveryHelper.sol             |      100 |    93.75 |      100 |      100 |                |
  ProtocolFeePercentagesProvider.sol |      100 |    97.22 |      100 |      100 |                |
  ProtocolFeeSplitter.sol            |    89.47 |    93.75 |    93.75 |    92.16 |156,157,159,207 |
  ProtocolFeesWithdrawer.sol         |      100 |    88.24 |      100 |      100 |                |
  ProtocolIdRegistry.sol             |      100 |      100 |      100 |      100 |                |
  SpotPriceOracle.sol                |      100 |    91.67 |      100 |      100 |                |
 contracts/relayer/                  |    93.02 |    83.52 |    93.75 |    93.35 |                |
  AaveWrapping.sol                   |      100 |    88.89 |      100 |      100 |                |
  BalancerRelayer.sol                |      100 |    66.67 |      100 |    94.74 |             74 |
  BaseRelayerLibrary.sol             |      100 |      100 |      100 |      100 |                |
  BaseRelayerLibraryCommon.sol       |      100 |      100 |      100 |      100 |                |
  CompoundV2Wrapping.sol             |      100 |       50 |      100 |      100 |                |
  ERC4626Wrapping.sol                |        0 |      100 |        0 |        0 |... 41,51,53,55 |
  EulerWrapping.sol                  |      100 |      100 |      100 |      100 |                |
  GaugeActions.sol                   |      100 |      100 |      100 |      100 |                |
  GearboxWrapping.sol                |      100 |      100 |      100 |      100 |                |
  IBaseRelayerLibrary.sol            |      100 |      100 |      100 |      100 |                |
  LidoWrapping.sol                   |      100 |      100 |      100 |      100 |                |
  ReaperWrapping.sol                 |      100 |      100 |      100 |      100 |                |
  SiloWrapping.sol                   |      100 |      100 |      100 |      100 |                |
  TetuWrapping.sol                   |      100 |      100 |      100 |      100 |                |
  UnbuttonWrapping.sol               |      100 |      100 |      100 |      100 |                |
  VaultActions.sol                   |    87.93 |       82 |    90.48 |    89.84 |... 426,427,431 |
  VaultPermit.sol                    |        0 |      100 |        0 |        0 |          39,52 |
  VaultQueryActions.sol              |    97.56 |    78.57 |      100 |    94.34 |      44,92,111 |
  YearnWrapping.sol                  |      100 |      100 |      100 |      100 |                |
 contracts/relayer/interfaces/       |      100 |      100 |      100 |      100 |                |
  IMockEulerProtocol.sol             |      100 |      100 |      100 |      100 |                |
 contracts/relayer/special/          |        0 |      100 |        0 |        0 |                |
  DoubleEntrypointFixRelayer.sol     |        0 |      100 |        0 |        0 |... 165,178,179 |
-------------------------------------|----------|----------|----------|----------|----------------|
All files                            |    73.39 |    74.61 |    81.19 |    71.95 |                |
-------------------------------------|----------|----------|----------|----------|----------------|
```

Changelog

- 2025-01-17 - Initial Report
- 2025-01-23 - Final Report

About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

Quantstamp