Report

v. 1.0

Customer

Uniswap

# Smart Contract Audit

# v4-periphery | Universal router

5th September 2024

Report prepared by

ABDK Consulting

# Contents


**1 Changelog** **5**

**2 Introduction** **6**

**3 Project scope** **7**

**4 Methodology** **9**

**5 Our findings** **10**

**6 Major Issues** **11**
CVF-1. Reported . . . . . 11
CVF-2. Reported . . . . . 11
CVF-3. Reported . . . . . 11
CVF-4. Reported . . . . . 12

**7 Moderate Issues** **13**
CVF-5. Reported . . . . . 13
CVF-6. Reported . . . . . 13
CVF-7. Reported . . . . . 13
CVF-8. Reported . . . . . 14
CVF-9. Reported . . . . . 14
CVF-10. Reported . . . . . 15
CVF-11. Reported . . . . . 15
CVF-12. Reported . . . . . 16
CVF-13. Reported . . . . . 16
CVF-14. Reported . . . . . 16
CVF-15. Reported . . . . . 17
CVF-16. Reported . . . . . 17
CVF-17. Reported . . . . . 17
CVF-18. Reported . . . . . 18
CVF-19. Reported . . . . . 18
CVF-20. Reported . . . . . 18

**8 Minor Issues** **19**
CVF-21. Reported . . . . . 19
CVF-22. Reported . . . . . 19
CVF-23. Reported . . . . . 20
CVF-24. Reported . . . . . 20
CVF-25. Reported . . . . . 20
CVF-26. Reported . . . . . 21
CVF-27. Reported . . . . . 21
CVF-28. Reported . . . . . 21
CVF-29. Reported . . . . . 21

CVF-30. Reported . . . . . 22
CVF-31. Reported . . . . . 22
CVF-32. Reported . . . . . 22
CVF-33. Reported . . . . . 23
CVF-34. Reported . . . . . 23
CVF-35. Reported . . . . . 23
CVF-36. Reported . . . . . 23
CVF-37. Reported . . . . . 24
CVF-38. Reported . . . . . 24
CVF-39. Reported . . . . . 24
CVF-40. Reported . . . . . 25
CVF-41. Reported . . . . . 25
CVF-42. Reported . . . . . 25
CVF-43. Reported . . . . . 26
CVF-44. Reported . . . . . 26
CVF-45. Reported . . . . . 26
CVF-46. Reported . . . . . 26
CVF-47. Reported . . . . . 27
CVF-48. Reported . . . . . 27
CVF-49. Reported . . . . . 27
CVF-50. Reported . . . . . 27
CVF-51. Reported . . . . . 28
CVF-52. Reported . . . . . 28
CVF-53. Reported . . . . . 28
CVF-54. Reported . . . . . 28
CVF-55. Reported . . . . . 29
CVF-56. Reported . . . . . 29
CVF-57. Reported . . . . . 29
CVF-58. Reported . . . . . 29
CVF-59. Reported . . . . . 30
CVF-60. Reported . . . . . 30
CVF-61. Reported . . . . . 30
CVF-62. Reported . . . . . 31
CVF-63. Reported . . . . . 31
CVF-64. Reported . . . . . 31
CVF-65. Reported . . . . . 32
CVF-66. Reported . . . . . 32
CVF-67. Reported . . . . . 32
CVF-68. Reported . . . . . 32
CVF-69. Reported . . . . . 33
CVF-70. Reported . . . . . 33
CVF-71. Reported . . . . . 33
CVF-72. Reported . . . . . 34
CVF-73. Reported . . . . . 34
CVF-74. Reported . . . . . 34
CVF-75. Reported . . . . . 34

CVF-76. Reported . . . . . 35
CVF-77. Reported . . . . . 35
CVF-78. Reported . . . . . 35
CVF-79. Reported . . . . . 35
CVF-80. Reported . . . . . 36
CVF-81. Reported . . . . . 36
CVF-82. Reported . . . . . 36
CVF-83. Reported . . . . . 37
CVF-84. Reported . . . . . 37
CVF-85. Reported . . . . . 37
CVF-86. Reported . . . . . 38
CVF-87. Reported . . . . . 38
CVF-88. Reported . . . . . 38
CVF-89. Reported . . . . . 38
CVF-90. Reported . . . . . 39
CVF-91. Reported . . . . . 39
CVF-92. Reported . . . . . 39
CVF-93. Reported . . . . . 39
CVF-94. Reported . . . . . 40
CVF-95. Reported . . . . . 40
CVF-96. Reported . . . . . 40
CVF-97. Reported . . . . . 40
CVF-98. Reported . . . . . 41
CVF-99. Reported . . . . . 41
CVF-100. Reported . . . . . 41
CVF-101. Reported . . . . . 41
CVF-102. Reported . . . . . 42
CVF-103. Reported . . . . . 42
CVF-104. Reported . . . . . 42
CVF-105. Reported . . . . . 43
CVF-106. Reported . . . . . 43
CVF-107. Reported . . . . . 43
CVF-108. Reported . . . . . 44
CVF-109. Reported . . . . . 44

# 1 Changelog

| # | Date | Author | Description |
|---|---|---|---|
| 0.1 | 05.09.24 | D. Khovratovich | Initial Draft |
| 0.2 | 05.09.24 | D. Khovratovich | Minor revision |
| 1.0 | 05.09.24 | D. Khovratovich | Release |

# 2 Introduction

**All modifications to this document are prohibited. Violators will be prosecuted to the full extent of the U.S. law.**

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

Uniswap V4, the latest iteration of the Uniswap protocol, is a significant advancement in decentralized exchanges (DEXs) and automated market makers (AMMs).

# 3 Project scope

We were asked to review:

- Periphery code
- Universal router code

Files:

| | | |
|---|---|---|
| **unirouter/base/** | | |
| Callbacks.sol | Dispatcher.sol | Lock.sol |
| **unirouter/libraries** | | |
| Locker.sol | | |
| **unirouter/modules/uniswap/v4/** | | |
| V4SwapRouter.sol | MigratorImmutables.sol | V3ToV4Migrator.sol |
| **unirouter/** | | |
| UniversalRouter.sol | | |
| **v4peri/base/** | | |
| BaseActionsRouter.sol | DeltaResolver.sol | EIP712_v4.sol |
| ERC721Permit_v4.sol | ImmutableState.sol | Multicall_v4.sol |
| Notifier.sol | Permit2Forwarder.sol | PoolInitializer.sol |
| ReentrancyLock.sol | SafeCallback.sol | UnorderedNonce.sol |
| **v4peri/interfaces/** | | |
| IEIP712_v4.sol | IERC721Permit_v4.sol | IMulticall_v4.sol |
| INotifier.sol | IPositionManager.sol | IQuoter.sol |
| ISubscriber.sol | IV4Router.sol | |
| **v4peri/lens/** | | |
| Quoter.sol | | |

| | | | |
|---|---|---|---|
| **v4peri/libraries/** | | | |
| | ActionConstants.sol | Actions.sol | BipsLibrary.sol |
| | CalldataDecoder.sol | ERC721PermitHash.sol | Locker.sol |
| | PathKey.sol | PoolTicksCounter.sol | PositionConfig.sol |
| | SafeCast.sol | SlippageCheck.sol | |
| **v4peri/** | | | |
| | PositionManager.sol | V4Router.sol | |

**All found issues were left as is.**
**After fixing the indicated issues, the smart contracts should be re-audited.**

# 4 Methodology

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

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

We classify issues by the following severity levels:

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

# 5 Our findings

We found 4 major, and a few less important issues.

# 6 Major Issues

## CVF-1 Reported

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

**Description** Underflow is possible when converting “int128” to “uint128”.

**Recommendation** Use safe conversion.

```
15 if (uint128(delta.amount0()) < amount0Min || uint128(delta.amount1()
   ↪ ) < amount1Min) {
```

```
23 if (uint128(-delta.amount0()) > amount0Max || uint128(-delta.amount1
   ↪ ()) > amount1Max) {
```

## CVF-2 Reported

- **Category** Procedural
- **Source** Locker.sol

**Description** Solidity permits junk in unused bits of narrow types.

**Recommendation** Explicitly clear unused buts of “locker” before storing.

```
14 tstore(LOCKER_SLOT, locker)
```

## CVF-3 Reported

- **Category** Flaw
- **Source** Lock.sol

**Description** There is no check to ensure “msg.value” isn’t zero, while zero address has a special meaning of “not locked”.

**Recommendation** Add an explicit check to ensure “msg.value” isn’t zero, or store an additional bit, along with a locker’s address, to signal locked state.

```
14 Locker.set(msg.sender);
```

## CVF-4 Reported

- **Category** Flaw
- **Source** Dispatcher.sol

**Description** This isn't enforced.

**Recommendation** Perform an appropriate selector check to enforce this.

```
258 // should only call modifyLiquidities() to mint or increase
       ↪ liquidity
```

# 7 Moderate Issues

## CVF-5 Reported

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

**Description** There is no check to ensure “actions” is within “bytes”.

**Recommendation** Implement proper bounds checks.

```
30 actions.length := calldataload(actionsPtr)
```

```
34 actions.offset := add(actionsPtr, 0x20)
```

## CVF-6 Reported

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

**Description** Underflow is possible here.

**Recommendation** Revert on underflow.

```
38 let relativeOffset := sub(params.offset, _bytes.offset)
```

```
259 relativeOffset := sub(offset, _bytes.offset)
```

## CVF-7 Reported

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

**Description** Overflow is possible here.

**Recommendation** Revert on overflow.

```
40 if lt(_bytes.length, add(params.length, relativeOffset)) {
```

## CVF-8 Reported

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

**Description** There are no checks to ensure “swapParams” are within “params”.

**Recommendation** Implement proper bounds checks.

```
122 swapParams := add(params.offset, calldataload(params.offset))
```

```
134 swapParams := add(params.offset, calldataload(params.offset))
```

```
146 swapParams := add(params.offset, calldataload(params.offset))
```

```
158 swapParams := add(params.offset, calldataload(params.offset))
```

## CVF-9 Reported

- **Category** Suboptimal
- **Source** PoolTicksCounter.sol

**Description** In other words, we don’t want to count the lowest of tickBefore and tickAfter, but we do want to count the highest of them.

**Recommendation** Use this fact to simplify the function: set tickFrom = min(tickBefore, tickAfter) + 1, tickTo = max(tickBefore, tickAfter), and then count bits from tickFrom to tickTo.

```
24 /// direction of the swap. If we are swapping upwards (tickAfter >
      ↪ tickBefore) we don't want to count tickBefore but we do
   /// want to count tickAfter. The opposite is true if we are swapping
      ↪  downwards.
```

## CVF-10 Reported

- **Category** Suboptimal
- **Source** PoolTicksCounter.sol

**Description** The mask logic is only needed only on the first and last iteration. Performing it on every iteration is inefficient.

**Recommendation** Refactor like this: if (cache.wordPosLower == cache.wordPosHigher) { // Count the only word taking into account cache.bitPosLower and cache.bit\PosHigher } else { // Count the first word taking into account cache.bitPosLower // Iterate through middle words and count them completely // Count the last word taking into account cache.bitPosHiger }

```
70  uint256 mask = type(uint256).max << cache.bitPosLower;
```

```
74      if (cache.wordPosLower == cache.wordPosHigher) {
            mask = mask & (type(uint256).max >> (255 - cache.
               ↪ bitPosHigher));
```

```
79      uint256 masked = bmLower & mask;
```

```
83      mask = type(uint256).max;
```

## CVF-11 Reported

- **Category** Suboptimal
- **Source** PoolTicksCounter.sol

**Description** This function is inefficient.

**Recommendation** Here is an efficient implementation: https://github.com/Vectorized/solady/blob/main/src/utils/LibBit.sol#L71-L82

```
97  function countOneBits(uint256 x) private pure returns (uint16) {
```

## CVF-12 Reported

- **Category** Unclear behavior
- **Source** IMulticall_v4.sol

**Description** It is unclear how a method could known whether it is called from “multicall”.

**Recommendation** Elaborate more on how to safely implement methods compatible with multicall.

```
8 /// @dev The `msg.value` should not be trusted for any method
    ↪ callable from multicall.
```

## CVF-13 Reported

- **Category** Procedural
- **Source** DeltaResolver.sol

**Recommendation** This should be surrounded with an “unchecked” block to save gas and to support the -(2^255) value.

```
59 amount = uint256(-_amount);
```

## CVF-14 Reported

- **Category** Flaw
- **Source** ERC721Permit_v4.sol

**Description** Such TODOs are quite dangerous.

**Recommendation** Pass the URI as a constructor argument. Note that short strings could be efficiently packed as bytes32 and stored in immutable variables: https://gist.github.com/3sGgpQ8H/567354534170905e047b299286697e19

```
104 // TODO: to be implemented after audits
function tokenURI(uint256) public pure override returns (string
    ↪ memory) {
     return "https://example.com";
}
```

## CVF-15 Reported

- **Category** Suboptimal
- **Source** Multicall_v4.sol

**Description** This function is payable, but there is no clear way to distribute the ether, provided along with the call among subcalls.

**Recommendation** Implement some mechanism for the subcalls to know how much incoming ether is not yet "consumed" by previously executed subcalls, and to "consume" this unconsumed ether.

```
10 function multicall(bytes[] calldata data) public payable override
   ↪ returns (bytes[] memory results) {
```

## CVF-16 Reported

- **Category** Procedural
- **Source** Multicall_v4.sol

**Description** Error message emitted here doesn't include if index of the failed subcallm which could make problem investigations harder.

**Recommendation** Wrap the raw message from a subcall into a error, and include ths subcall index into this error as an extra parameter.

```
18 revert(add(result, 0x20), mload(result))
```

## CVF-17 Reported

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

**Description** There is no check to ensure "_subscriber" is not zero, i.e. that a subscription for given token actually exists.

**Recommendation** Explicitly require "_subscriber" to be not zero.

```
47 ISubscriber _subscriber = subscriber[tokenId];
```

## CVF-18 Reported

- **Category** Suboptimal
- **Source** ReentrancyLock.sol

**Description** This cannot properly handle situation when “msg.sender” is zero. While this is very unlikely at the current time, future Ethereum updates could make this possible.

**Recommendation** Explicitly require “msg.sender” not not be zero or properly handle zero “msg.sender”, e.g. by storing an extra bit along with the locker’s address in the same storage slot.

```
12 Locker.set(msg.sender);
```

## CVF-19 Reported

- **Category** Procedural
- **Source** PositionManager.sol

**Description** There is no “_beforeModify” function anywhere around, nor calls to it.

**Recommendation** Remove this comment.

```
254 // _beforeModify is not called here because the tokenId is newly
    ↪ minted
```

## CVF-20 Reported

- **Category** Procedural
- **Source** Lock.sol

**Description** According to Solidity documentation (https://docs.solidity-lang.org/en/v0.8.26/contracts.html#modifiers): “The _ symbol can appear in the modifier multiple times. Each occurrence is replaced with the function body, and the function returns the return value of the final occurrence.” So using “_” twice in a modifier effectively duplicates the function code.

**Recommendation** Refactor the modifier to use “_” only once.

```
15 _;
```

```
19 _;
```

# 8 Minor Issues

## CVF-21 Reported

- **Category** Procedural
- **Source** PositionConfig.sol

**Description** Consider specifying as "^0.8.0" unless there is something special regarding this particular version.

**Recommendation** Also relevant for: PathKey.sol, Actions.sol, ActionConstants.sol, ERC721PermitHash.sol, BipsLibrary.sol, Locker.sol, PoolTicksCounter.sol, Quoter.sol, IV4Router.sol, IMulticall_v4.sol, ISubscriber.sol, INotifier.sol, IPositionManager.sol, IQuoter.sol, ImmutableState.sol, SafeCallback.sol,BaseActionsRouter.sol, DeltaResolver.sol, EIP712_v4.sol, UnorderedNonce.sol, ERC721Permit_v4.sol,Multicall_v4.sol, Notifier.sol, Permit2Forwarder.sol, PoolInitializer.sol, ReentrancyLock.sol, V4Router.sol, PositionManager.sol, V4SwapRouter.sol, V3ToV4Migrator.sol, MigratorImmutables.sol, Locker.sol, Callbacks.sol, Lock.sol, Dispatcher.sol, UniversalRouter.sol.

```
2 pragma solidity ^0.8.24;
```

## CVF-22 Reported

- **Category** Procedural
- **Source** PositionConfig.sol

**Description** Declaring a top-level structure in a file named after a library makes it harder navigating through code.

**Recommendation** Put the structure declaration into the library or move it into a separate file.

```
7 struct PositionConfig {
```

## CVF-23 Reported

- **Category** Suboptimal
- **Source** PositionConfig.sol

**Description** Is the assembly block below really significantly more efficient than the hash expression in this comment?

**Recommendation** Replace assembly block with hash expression, if there are no significant benefit in using assembly block.

```
57 // id = keccak256(abi.encodePacked(currency0, currency1, fee,
      ↪ tickSpacing, hooks, tickLower, tickUpper))) >> 1
```

## CVF-24 Reported

- **Category** Suboptimal
- **Source** PositionConfig.sol

**Description** This code is redundant, as Solidity permits junk in free memory https://-docs.soliditylang.org/en/v0.8.26/internals/layout_in_memory.html

**Recommendation** Remove this code.

```
70 // now clean the memory we used
mstore(add(fmp, 0x40), 0) // fmp+0x40 held hooks (14 bytes),
   ↪ tickLower, tickUpper
mstore(add(fmp, 0x20), 0) // fmp+0x20 held currency1, fee,
   ↪ tickSpacing, hooks (6 bytes)
mstore(fmp, 0) // fmp held currency0
```

## CVF-25 Reported

- **Category** Procedural
- **Source** PathKey.sol

**Description** This version requirement is inconsistent with other files in this code base.

**Recommendation** Use consistent version requirements across the code base.

```
2 pragma solidity ^0.8.20;
```

## CVF-26 Reported

- **Category** Procedural
- **Source** PathKey.sol

**Description** Declaring a top-level structure in a file named after a library makes it harder navigating through code.

**Recommendation** Put the structure declaration into the library or move it into a separate file.

```
8 struct PathKey {
```

## CVF-27 Reported

- **Category** Procedural
- **Source** CalldataDecoder.sol

**Description** This version requirement is inconsistent with other files in this code base.

**Recommendation** Use consistent version requirements across the code base.

```
2 pragma solidity ^0.8.0;
```

## CVF-28 Reported

- **Category** Suboptimal
- **Source** CalldataDecoder.sol

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

```
12 error SliceOutOfBounds();
```

## CVF-29 Reported

- **Category** Procedural
- **Source** Actions.sol

**Recommendation** This library could be turned into a enum.

```
6 library Actions {
```

## CVF-30 Reported

- **Category** Procedural
- **Source** ActionConstants.sol

**Description** This version requirement is inconsistent with other files in this code base.

**Recommendation** Use consistent version requirements across the code base.

```
2 pragma solidity ^0.8.19;
```

## CVF-31 Reported

- **Category** Procedural
- **Source** ActionConstants.sol

**Description** This comment is useless, as it just describes the constant value.

**Recommendation** Change the comment to describe the role of the constant.

```
8 /// This value is equivalent to 1<<255, i.e. a singular 1 in the
     ↪ most significant bit.
```

## CVF-32 Reported

- **Category** Procedural
- **Source** ERC721PermitHash.sol

**Description** Solidity compiler is smart enough to precompute constant hash expressions.

**Recommendation** Use the hash expression instead of a hardcoded hash value.

```
5 /// @dev Value is equal to keccak256("Permit(address spender,uint256
     ↪  tokenId,uint256 nonce,uint256 deadline)");
bytes32 constant PERMIT_TYPEHASH =
0x49ecf333e5b8c95c40fdafc95c1ad136e8914a8fb55e9dc8bb01eaa83a2df9ad;
```

```
8 /// @dev Value is equal to keccak256("PermitForAll(address operator,
     ↪ bool approved,uint256 nonce,uint256 deadline)");
bytes32 constant PERMIT_FOR_ALL_TYPEHASH =
10 0x6673cb397ee2a50b6b8401653d3638b4ac8b3db9c28aa6870ffceb7574ec2f76;
```

## CVF-33 Reported

- **Category** Procedural
- **Source** BipsLibrary.sol

**Description** This version requirement is inconsistent with other files in this code base.

**Recommendation** Use consistent version requirements across the code base.

```
2 pragma solidity ^0.8.19;
```

## CVF-34 Reported

- **Category** Suboptimal
- **Source** BipsLibrary.sol

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

```
10 error InvalidBips();
```

## CVF-35 Reported

- **Category** Procedural
- **Source** BipsLibrary.sol

**Recommendation** Brackets around multiplication are redundant.

```
16 return (amount * bips) / BPS_DENOMINATOR;
```

## CVF-36 Reported

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

**Description** Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, while some intermediary calculation overflows.

**Recommendation** Prevent phantom overflow is some way, for example in a way described here: https://medium.com/coinmonks/math-in-solidity-part-3-percents-and-proportions-4db014e080b1#4821

```
16 return (amount * bips) / BPS_DENOMINATOR;
```

## CVF-37 Reported

- **Category** Procedural
- **Source** Locker.sol

**Description** Solidity compiler is smart enough to precompute constant hash expressions.

**Recommendation** Use the hash expression instead of a hardcoded hash value.

```
7 // The slot holding the locker state, transiently. bytes32(uint256(
     ↪ keccak256("LockedBy")) - 1)
  bytes32 constant LOCKED_BY_SLOT =
  0x0aedd6bde10e3aa2adec092b02a3e3e805795516cda41f27aa145b8f300af87a;
```

## CVF-38 Reported

- **Category** Procedural
- **Source** PoolTicksCounter.sol

**Description** This version requirement is inconsistent with other files in this code base.

**Recommendation** Use consistent version requirements across the code base.

```
2 pragma solidity >=0.8.20;
```

## CVF-39 Reported

- **Category** Procedural
- **Source** PoolTicksCounter.sol

**Description** The value “tickBefore / key.tickSpacing” is calculated twice.

**Recommendation** Calculate once and reuse.

```
35 int16 wordPos = int16((tickBefore / key.tickSpacing) >> 8);
   uint8 bitPos = uint8(uint24((tickBefore / key.tickSpacing) % 256));
```

## CVF-40 Reported

- **Category** Suboptimal
- **Source** PoolTicksCounter.sol

**Recommendation** Bitwise “AND” would be more efficient than reminder operator.

```
36 uint8 bitPos = uint8(uint24((tickBefore / key.tickSpacing) % 256));
```

```
39 uint8 bitPosAfter = uint8(uint24((tickAfter / key.tickSpacing) %
   ↪ 256));
```

## CVF-41 Reported

- **Category** Procedural
- **Source** PoolTicksCounter.sol

**Description** The value “tickAfter / key.tickSpacing” is calculated twice.

**Recommendation** Calculate once and reuse.

```
38 int16 wordPosAfter = int16((tickAfter / key.tickSpacing) >> 8);
   uint8 bitPosAfter = uint8(uint24((tickAfter / key.tickSpacing) %
   ↪ 256));
```

## CVF-42 Reported

- **Category** Procedural
- **Source** SafeCast.sol

**Description** This version requirement is inconsistent with other files in this code base.

**Recommendation** Use consistent version requirements across the code base.

```
2 pragma solidity ^0.8.0;
```

## CVF-43 Reported

- **Category** Procedural
- **Source** SafeCast.sol

**Description** Such TODOs are quite dangerous.

**Recommendation** Don't forget to resolve it.

```
8 /// TODO after audits move this function to core's SafeCast.sol!
```

## CVF-44 Reported

- **Category** Procedural
- **Source** SafeCast.sol

**Description** The library name doesn't match the file name.

**Recommendation** Rename ether the library of the file.

```
9 library SafeCastTemp {
```

## CVF-45 Reported

- **Category** Procedural
- **Source** SlippageCheck.sol

**Description** This version requirement is inconsistent with other files in this code base.

**Recommendation** Use consistent version requirements across the code base.

```
2 pragma solidity ^0.8.0;
```

## CVF-46 Reported

- **Category** Procedural
- **Source** SlippageCheck.sol

**Description** The library name doesn't match the file name.

**Recommendation** Remane either the library of the file.

```
8 library SlippageCheckLibrary {
```

## CVF-47 Reported

- **Category** Suboptimal
- **Source** SlippageCheck.sol

**Recommendation** These errors could be made more useful be adding certain parameters into them.

```
9  error MaximumAmountExceeded();
10 error MinimumAmountInsufficient();
```

## CVF-48 Reported

- **Category** Procedural
- **Source** SlippageCheck.sol

**Description** The expressions “delta.amount0()” and “delta.amount1()” are are calculated twice.

**Recommendation** Calculate once and reuse.

```
33 delta.amount0() < 0 && amount0Max < uint128(-delta.amount0())
       || delta.amount1() < 0 && amount1Max < uint128(-delta.amount1())
```

## CVF-49 Reported

- **Category** Procedural
- **Source** Quoter.sol

**Description** This version requirement is inconsistent with other files in this code base.

**Recommendation** Use consistent version requirements across the code base.

```
2 pragma solidity ^0.8.20;
```

## CVF-50 Reported

- **Category** Procedural
- **Source** Quoter.sol

**Recommendation** This import should be grouped together with other Uniswap imports.

```
16 import {StateLibrary} from "@uniswap/v4-core/src/libraries/
     ↪ StateLibrary.sol";
```

## CVF-51 Reported

- **Category** Procedural
- **Source** Quoter.sol

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

```
55 constructor(IPoolManager _poolManager) SafeCallback(_poolManager) {}
```

## CVF-52 Reported

- **Category** Suboptimal
- **Source** Quoter.sol

**Recommendation** Just assign “params.exactAmount” to “cache.prevAmount” before the loop, to make this ternary operator unnecessary.

```
176 -int256(int128(i == 0 ? params.exactAmount : cache.prevAmount)),
```

## CVF-53 Reported

- **Category** Procedural
- **Source** IV4Router.sol

**Description** This version requirement is inconsistent with other files in this code base.

**Recommendation** Use consistent version requirements across the code base.

```
2 pragma solidity ^0.8.19;
```

## CVF-54 Reported

- **Category** Suboptimal
- **Source** IV4Router.sol

**Recommendation** These errors could be made more useful by adding certain parameters into them.

```
12 error V4TooLittleReceived();
```

```
14 error V4TooMuchRequested();
```

## CVF-55 Reported

- **Category** Procedural
- **Source** IEIP712_v4.sol

**Description** This version requirement is inconsistent with other files in this code base.

**Recommendation** Use consistent version requirements across the code base.

```
2 pragma solidity ^0.8.0;
```

## CVF-56 Reported

- **Category** Procedural
- **Source** IMulticall_v4.sol

**Description** This version requirement is inconsistent with other files in this code base.

**Recommendation** Use consistent version requirements across the code base.

```
2 pragma solidity ^0.8.19;
```

## CVF-57 Reported

- **Category** Suboptimal
- **Source** IPositionManager.sol

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

```
12 error DeadlinePassed();
```

## CVF-58 Reported

- **Category** Procedural
- **Source** IERC721Permit_v4.sol

**Description** Specifying a compiler version range without an upper bound is a bad practice, as it is impossible to guarantee compatibility with future major releases.

**Recommendation** Specify as “^0.7.0 || ^0.8.0’, or as “^0.7.5 || ^0.8.0” is there is something special regarding this particular version.

```
2 pragma solidity >=0.7.5;
```

## CVF-59 Reported

- **Category** Procedural
- **Source** IERC721Permit_v4.sol

**Description** This version requirement is inconsistent with other files in this code base.

**Recommendation** Use consistent version requirements across the code base.

```
2 pragma solidity >=0.7.5;
```

## CVF-60 Reported

- **Category** Procedural
- **Source** IQuoter.sol

**Description** This version requirement is inconsistent with other files in this code base.

**Recommendation** Use consistent version requirements across the code base.

```
2 pragma solidity ^0.8.20;
```

## CVF-61 Reported

- **Category** Suboptimal
- **Source** IQuoter.sol

**Recommendation** These errors could be made more useful by adding certain parameters into them.

```
14 error InvalidLockCaller();
   error InvalidQuoteBatchParams();
   error InsufficientAmountOut();
```

## CVF-62 Reported

- **Category** Suboptimal
- **Source** IQuoter.sol

**Recommendation** It would be more efficient to return a single array of structs with three fields, rather than three parallel arrays.

```
65 int128[] memory deltaAmounts,
   uint160[] memory sqrtPriceX96AfterList,
   uint32[] memory initializedTicksLoadedList
```

```
95 int128[] memory deltaAmounts,
   uint160[] memory sqrtPriceX96AfterList,
   uint32[] memory initializedTicksLoadedList
```

## CVF-63 Reported

- **Category** Procedural
- **Source** ImmutableState.sol

**Description** This version requirement is inconsistent with other files in this code base.

**Recommendation** Use consistent version requirements across the code base.

```
2 pragma solidity ^0.8.19;
```

## CVF-64 Reported

- **Category** Procedural
- **Source** SafeCallback.sol

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

```
11 constructor(IPoolManager _poolManager) ImmutableState(_poolManager)
     ↪ {}
```

## CVF-65 Reported

- **Category** Suboptimal
- **Source** SafeCallback.sol

**Recommendation** The “by” preposition looks redundant here, as “onlyPoolManager” would sound more conventional.

```
13 modifier onlyByPoolManager() {
```

## CVF-66 Reported

- **Category** Suboptimal
- **Source** BaseActionsRouter.sol

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

```
16 error InputLengthMismatch();
```

## CVF-67 Reported

- **Category** Procedural
- **Source** BaseActionsRouter.sol

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

```
21 constructor(IPoolManager _poolManager) SafeCallback(_poolManager) {}
```

## CVF-68 Reported

- **Category** Suboptimal
- **Source** BaseActionsRouter.sol

**Recommendation** Explicit conversion to “uint256” is redundant.

```
44 uint256 action = uint256(uint8(actions[actionIndex]));
```

## CVF-69 Reported

- **Category** Readability
- **Source** DeltaResolver.sol

**Recommendation** Should be “else return”.

```
79 return amount;
```

```
87 return amount;
```

```
101 return amount;
```

## CVF-70 Reported

- **Category** Procedural
- **Source** EIP712_v4.sol

**Description** This version requirement is inconsistent with other files in this code base.

**Recommendation** Use consistent version requirements across the code base.

```
2 pragma solidity ^0.8.20;
```

## CVF-71 Reported

- **Category** Procedural
- **Source** EIP712_v4.sol

**Description** Solidity compiler is smart enough to precompute constant hash expressions.

**Recommendation** Use the hash expression instead of a hardcoded hash value.

```
18 /// @dev equal to keccak256("EIP712Domain(string name,uint256
   ↪ chainId,address verifyingContract)")
19 bytes32 private constant _TYPE_HASH =
20 0x8cad95687ba82c2ce50e74f7b754645e5117c3a5bec8151c0726d5857980a866;
```

## CVF-72 Reported

- **Category** Procedural
- **Source** ERC721Permit_v4.sol

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

```
4 import {ERC721} from "solmate/src/tokens/ERC721.sol";
```

## CVF-73 Reported

- **Category** Procedural
- **Source** ERC721Permit_v4.sol

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

```
18 constructor(string memory name_, string memory symbol_) ERC721(name_
   ↪ , symbol_) EIP712_v4(name_) {}
```

## CVF-74 Reported

- **Category** Documentation
- **Source** ERC721Permit_v4.sol

**Description** It is unclear what this comment is about.

**Recommendation** Remove or rephrase the comment.

```
77 /// The zero address indicates there is no approved address
```

## CVF-75 Reported

- **Category** Suboptimal
- **Source** ERC721Permit_v4.sol

**Description** This function is too simple to be extracted.

**Recommendation** Remove the function and inline the check.

```
100 function _checkNoSelfPermit(address owner, address permitted)
    ↪ internal pure {
     if (owner == permitted) revert NoSelfPermit();
```

## CVF-76 Reported

- **Category** Procedural
- **Source** Multicall_v4.sol

**Description** This version requirement is inconsistent with other files in this code base.

**Recommendation** Use consistent version requirements across the code base.

```
2 pragma solidity ^0.8.19;
```

## CVF-77 Reported

- **Category** Procedural
- **Source** Notifier.sol

**Recommendation** This TODO should be resolved or removed.

```
11 /// TODO: Use CustomRevert library when it supports subcontext's
   ↪ addresss
```

## CVF-78 Reported

- **Category** Suboptimal
- **Source** Notifier.sol

**Recommendation** This error could be made more useful by adding the token ID as a parameter.

```
16 error AlreadySubscribed(address subscriber);
```

## CVF-79 Reported

- **Category** Procedural
- **Source** Notifier.sol

**Recommendation** The parameters of these events should be indexed.

```
18 event Subscribed(uint256 tokenId, address subscriber);
   event Unsubscribed(uint256 tokenId, address subscriber);
```

## CVF-80 Reported

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

**Recommendation** Events are usually named via nouns, such as “Subscription” or “Unsubscription”.

```
18 event Subscribed(uint256 tokenId, address subscriber);
   event Unsubscribed(uint256 tokenId, address subscriber);
```

## CVF-81 Reported

- **Category** Readability
- **Source** Notifier.sol

**Recommendation** This value could be rendered as “0.01e4”.

```
26 uint256 private constant BLOCK_LIMIT_BPS = 100;
```

## CVF-82 Reported

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

**Recommendation** The type for the “newSubscriber” argument should be “ISubscriber”.

```
30 function _subscribe(uint256 tokenId, PositionConfig memory config,
      ↪ address newSubscriber, bytes memory data)
```

## CVF-83 Reported

- **Category** Procedural
- **Source** Notifier.sol

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

```
38 try ISubscriber(newSubscriber).notifySubscribe(tokenId, config, data
   ↪ ) {}
```

```
51 try _subscriber.notifyUnsubscribe{gas: subscriberGasLimit}(tokenId,
   ↪ config, data) {} catch {}
```

```
59 try _subscriber.notifyModifyLiquidity(tokenId, config,
   ↪ liquidityChange) {}
```

```
67 try _subscriber.notifyTransfer(tokenId, previousOwner, newOwner) {}
```

## CVF-84 Reported

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

**Description** This event is emitted even if there were no subscription.

```
54 emit Unsubscribed(tokenId, address(_subscriber));
```

## CVF-85 Reported

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

**Description** The semantics of the returned values is unclear.

**Recommendation** Give a descriptive name to the returned value and/or explain in a documentation comment.

```
11 returns (int24)
```

## CVF-86 Reported

- **Category** Procedural
- **Source** V4Router.sol

**Description** This version requirement is inconsistent with other files in this code base.

**Recommendation** Use consistent version requirements across the code base.

```
2 pragma solidity ^0.8.19;
```

## CVF-87 Reported

- **Category** Procedural
- **Source** V4Router.sol

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

```
31 constructor(IPoolManager _poolManager) BaseActionsRouter(
   ↪ _poolManager) {}
```

## CVF-88 Reported

- **Category** Procedural
- **Source** PositionManager.sol

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

```
14 import {SafeTransferLib} from "solmate/src/utils/SafeTransferLib.sol
   ↪ ";
   import {ERC20} from "solmate/src/tokens/ERC20.sol";
```

## CVF-89 Reported

- **Category** Suboptimal
- **Source** PositionManager.sol

**Recommendation** This variable doesn't need to be public.

```
56 uint256 public nextTokenId = 1;
```

## CVF-90 Reported

- **Category** Procedural
- **Source** PositionManager.sol

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

```
64 {}
```

## CVF-91 Reported

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

**Description** When called not under lock, this function silently returns zero address.

**Recommendation** Revert or return the real message sender when called not under lock.

```
133 function msgSender() public view override returns (address) {
        return _getLocker();
```

## CVF-92 Reported

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

**Description** This function should return the minted token ID.

```
238 function _mint(
```

## CVF-93 Reported

- **Category** Suboptimal
- **Source** PositionManager.sol

**Recommendation** The type conversion is redundant here, as Solidity compiler would anyway do it automatically.

```
268 uint256 liquidity = uint256(getPositionLiquidity(tokenId, config));
```

## CVF-94 Reported

- **Category** Procedural
- **Source** PositionManager.sol

**Description** The comment doesn't add any information.

**Recommendation** Remove the comment.

```
278 // Burn the token.
    _burn(tokenId);
```

## CVF-95 Reported

- **Category** Procedural
- **Source** PositionManager.sol

**Recommendation** This TODO should be either resolved or removed.

```
353 // TODO: currency is guaranteed to not be eth so the native check in
       ↪  transfer is not optimal.
```

## CVF-96 Reported

- **Category** Procedural
- **Source** V4SwapRouter.sol

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

```
4 import {UniswapImmutables} from '../UniswapImmutables.sol';
  import {Permit2Payments} from '../../Permit2Payments.sol';
```

## CVF-97 Reported

- **Category** Bad datatype
- **Source** V4SwapRouter.sol

**Recommendation** The argument type should be "IPoolManager".

```
12 constructor(address _poolManager) V4Router(IPoolManager(_poolManager
      ↪ )) {}
```

## CVF-98 Reported

- **Category** Procedural
- **Source** MigratorImmutables.sol

**Description** Declaring a top-level struct in a file named after a contract makes it harder navigating through code.

**Recommendation** Put the struct declaration into the contract or move it into a separate file.

```
7 struct MigratorParameters {
```

## CVF-99 Reported

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

**Recommendation** The type for this field should be “INonfungiblePositionManager”.

```
8 address v3PositionManager;
```

## CVF-100 Reported

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

**Recommendation** The type for this field should be “IPositionManager”.

```
9 address v4PositionManager;
```

## CVF-101 Reported

- **Category** Procedural
- **Source** Locker.sol

**Description** Solidity compiler is smart enough to precompute constant hash expressions.

**Recommendation** Use the hash expression instead of a hardcoded hash value.

```
8 // The slot holding the locker state, transiently. bytes32(uint256(
    ↪ keccak256("Locker")) - 1)
bytes32 constant LOCKER_SLOT =
10 0x0e87e1788ebd9ed6a7e63c70a374cd3283e41cad601d21fbe27863899ed4a708;
```

## CVF-102 Reported

- **Category** Documentation
- **Source** Callbacks.sol

**Description** This contract actually implements functionality defined by a particular ERC, namely ERC-165.

**Recommendation** Rephrase comment.

```
6 /// @title ERC Callback Support
  /// @notice Implements various functions introduced by a variety of
     ↪ ERCs for security reasons.
  /// All are called by external contracts to ensure that this
     ↪ contract safely supports the ERC in question.
```

## CVF-103 Reported

- **Category** Procedural
- **Source** Dispatcher.sol

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

```
4 import {V2SwapRouter} from '../modules/uniswap/v2/V2SwapRouter.sol';
  import {V3SwapRouter} from '../modules/uniswap/v3/V3SwapRouter.sol';
```

```
7 import {BytesLib} from '../modules/uniswap/v3/BytesLib.sol';
  import {Payments} from '../modules/Payments.sol';
  import {PaymentsImmutables} from '../modules/PaymentsImmutables.sol'
     ↪ ;
```

```
12 import {Commands} from '../libraries/Commands.sol';
```

## CVF-104 Reported

- **Category** Procedural
- **Source** Dispatcher.sol

**Recommendation** This file could be imported as "./Callbacks.sol".

```
11 import {Callbacks} from '../base/Callbacks.sol';
```

## CVF-105 Reported

- **Category** Suboptimal
- **Source** Dispatcher.sol

**Description** Conversion from “bytes1” to “uint8” involves a shift operation.

**Recommendation** Consider passing the comment type as “uint8” to avoid shifting.

36
```
uint256 command = uint8(commandType & Commands.COMMAND_TYPE_MASK);
```

## CVF-106 Reported

- **Category** Suboptimal
- **Source** Dispatcher.sol

**Description** Indentation doesn’t suggest, that the comment belongs to the “else” statement below.

**Recommendation** Put the comment after the “else” statement like this: } else { // 008 <= command < 0x10

135
```
        // 0x08 <= command < 0x10
    } else {
```

220
```
    // 0x10 <= command < 0x18
} else {
```

## CVF-107 Reported

- **Category** Procedural
- **Source** Dispatcher.sol

**Recommendation** This check should be done earlier, before obtaining the token ID.

245
```
if (!isValidAction(selector)) {
```

## CVF-108 Reported

- **Category** Procedural
- **Source** UniversalRouter.sol

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

```
6  import {RouterParameters} from './base/RouterImmutables.sol';
   import {PaymentsImmutables, PaymentsParameters} from './modules/
      ↪ PaymentsImmutables.sol';
   import {UniswapImmutables, UniswapParameters} from './modules/
      ↪ uniswap/UniswapImmutables.sol';

10 import {Commands} from './libraries/Commands.sol';
   import {IUniversalRouter} from './interfaces/IUniversalRouter.sol';
```

## CVF-109 Reported

- **Category** Procedural
- **Source** UniversalRouter.sol

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

```
27 {}
```

ABDK
Consulting

## About us

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

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

## Contact


**Email**

dmitry@abdkconsulting.com

**Website**

abdk.consulting

**Twitter**

twitter.com/ABDKconsulting

**LinkedIn**

linkedin.com/company/abdk-consulting