Report

v. 1.0

Customer

Enso

Smart Contract Audit

Enso Router

9th July 2025

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	09.07.25	A. Zveryanskaya	Initial Draft
0.2	09.07.25	A. Zveryanskaya	Minor revision
1.0	09.07.25	A. Zveryanskaya	Release

2 Introduction

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

A network that encompasses all blockchains and smart contracts into one network, enabling developers to focus solely on their product, community, and distribution rather than the intricacies of blockchain development by integrating 1 tool: Enso.

3 Project scope

We were asked to review:

- Original Code
- Code with Fixes

Files:

```
/
    EnsoRouter.sol          EnsoShortcuts.sol          StargateV2Receiver.sol
```

4 Methodology

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

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

We classify issues by the following severity levels:

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

5 Our findings

We provided the client with some recommendations.

Moderate	Info	Fixed
	2	4

Fixed 4 out of 6 issues

6 Moderate Issues

CVF-2 INFO

- **Category** Flaw
- **Source** EnsoRouter.sol

Description Estimating the output amount by analysing how much the recipient balance was changed is unreliable, as the recipient may execute some logic when receiving assets. For example, the recipient may forward the received assets synchronously, or obtain more of the same assets from other sources.

Recommendation Refactor the code to properly count the amount of assets sent to the recipient.

Client Comment *We acknowledge there is an edge case where a ERC-777 token could be used with a receiver address that has implemented a callback that changes their balance. While it is an option to first send tokens to a contract we control to then check balances before any callback can be triggered, it would exclude us from using protocols that use non-transferrable tokens but support passing a receiver address (and would also increase gas costs to the user). Since the benefits of making this change are limited to supporting this edge case at the expense of all users, we are choosing not to change this functionality. For those sophisticated users who have implemented a callback in their receiver address, we can reasonably expect that they can handle their own amount out checks in the weiroll script or in their callback function and so recommend they use 'routeSingle' or 'routeMulti' instead.*

194
```
uint256 amountOut = balance - prevBalance;
if (amountOut < minAmountOut) revert AmountTooLow(token, amountOut,
   ↪ minAmountOut);
```

CVF-3 FIXED

- **Category** Bad datatype
- **Source** StargateV2Receiver.sol

Description The same types are declared in the "EnsoRouter.sol" file.

Recommendation Refactor the code to avoid duplication.

Client Comment *Fixed. Although we don't agree with the severity. This is a minor issue.*

11
```
enum TokenType {
```

18
```
struct Token {
```

CVF-4 FIXED

- **Category** Bad naming
- **Source** StargateV2Receiver.sol

Description The semantics of this function is confusing. The name suggests it should return multiple IDs, while is actually returns a single “int16” value. The semantics of the argument is also unclear.

Recommendation Explain the function semantics in a documentation comment and give descriptinve names to the argument and the returned value.

Client Comment *This is a Stargate function on a contract we are interfacing with and so we have no control over the naming choices. We included a comment in the contract explaining the purpose of this call.*

```
33 function assetIds(address) external view returns (uint16);
```

CVF-5 FIXED

- **Category** Procedural
- **Source** StargateV2Receiver.sol

Description The original error message is dropped here.

Recommendation Include the original error message into the error.

```
87 } catch {
       // if shortcut fails send funds to receiver
       emit ShortcutExecutionFailed(_guid);
```

```
124 if (!success) revert TransferFailed(receiver);
```

CVF-6 FIXED

- **Category** Suboptimal
- **Source** EnsoRouter.sol

Description This check makes the “amount” value redundant.

Recommendation Consider using empty token.data and just using the whole “msg.value” or even “address(this).balance”.

Client Comment *Removed this check and no longer attempt to decode data for the native asset*

```
137 if (msg.value != amount) revert WrongMsgValue(msg.value, amount);
```

CVF-7 INFO

- **Category** Procedural
- **Source** EnsoShortcuts.sol

Description This function seems to play a crucial role in security, however, we didn’t review the usages of this function.

```
15 function _checkMsgSender() internal view override {
```

7 Recommendations

CVF-8 FIXED

- **Category** Procedural
- **Source** StargateV2Receiver.sol

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

Recommendation Also relevant for: EnsoRouter.sol, EnsoShortcuts.sol.

Client Comment *Changed to ^0.8.20 and ^0.8.24 which are the lowest versions supported by dependencies*

```
2 pragma solidity ^0.8.28;
```

CVF-9 INFO

- **Category** Procedural
- **Source** StargateV2Receiver.sol

Description We didn’t review these files.

```
4 import { OFTComposeMsgCodec } from "@layerzerolabs/lz-evm-oapp-v2/
    ↪ contracts/oft/libs/OFTComposeMsgCodec.sol";
  import { ILayerZeroComposer } from "@layerzerolabs/lz-evm-protocol-
    ↪ v2/contracts/interfaces/ILayerZeroComposer.sol";
```

CVF-10 FIXED

- **Category** Procedural
- **Source** StargateV2Receiver.sol

Recommendation This interface should be moved into a separate file named “IRouter.sol”.

```
10 interface IRouter {
```

CVF-11 FIXED

- **Category** Procedural
- **Source** StargateV2Receiver.sol

Recommendation This interface should be moved into a separate file named “IToken-Messaging.sol”.

```
32 interface ITokenMessaging {
```

CVF-12 FIXED

- **Category** Procedural
- **Source** StargateV2Receiver.sol

Recommendation This interface should be moved into a separate file named “IPool.sol”.

```
36 interface IPool {
```

CVF-13 INFO

- **Category** Bad datatype
- **Source** StargateV2Receiver.sol

Recommendation The return type should be more specific.

Client Comment *Disagree.*

```
37 function token() external view returns (address);
```

CVF-14 INFO

- **Category** Bad datatype
- **Source** StargateV2Receiver.sol

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

Client Comment *Disagree.*

```
44 address private constant _NATIVE_ASSET = address(0);
```

CVF-15 INFO

- **Category** Bad naming
- **Source** StargateV2Receiver.sol

Recommendation Events are usually named via nouns, such as “ShortcutExecution” or “ShortcutExecutionFailure”.

Client Comment *Disagree.*

```
52 event ShortcutExecutionSuccessful(bytes32 guid);
   event ShortcutExecutionFailed(bytes32 guid);
```

CVF-16 FIXED

- **Category** Suboptimal
- **Source** StargateV2Receiver.sol

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

```
59 error InvalidAsset();
```

CVF-17 INFO

- **Category** Bad datatype
- **Source** StargateV2Receiver.sol

Recommendation The type for the “_tokenMessaging” argument should be “ITokenMessaging”.

Client Comment *Disagree. I’d rather the constructor to be easily ingested by test or deployment scripts. I’d rather not have to unnecessarily import otherwise unused interface files just to deploy a contract.*

```
61 constructor(address _endpoint, address _tokenMessaging, address
   ↪ _router, address _owner, uint256 _reserveGas) Ownable(_owner)
   ↪ {
```

CVF-18 INFO

- **Category** Bad datatype
- **Source** StargateV2Receiver.sol

Recommendation The type for the "_router" argument should be "IRouter".

Client Comment *Same as above.*

```
61 constructor(address _endpoint, address _tokenMessaging, address
   ↪ _router, address _owner, uint256 _reserveGas) Ownable(_owner)
   ↪ {
```

CVF-19 INFO

- **Category** Unclear behavior
- **Source** StargateV2Receiver.sol

Description There is no range check for the "_reserveGas" argument.

Recommendation Add an appropriate check.

Client Comment *Disagree. The acceptable upper limit is the chain's block gaslimit and there is no consistency between chains. Any other value would be arbitrary. We are capable of setting a reasonable value in our deployment script without on-chain restrictions*

```
61 constructor(address _endpoint, address _tokenMessaging, address
   ↪ _router, address _owner, uint256 _reserveGas) Ownable(_owner)
   ↪ {
```

CVF-20 FIXED

- **Category** Bad naming
- **Source** StargateV2Receiver.sol

Description The semantics of the last two arguments in unclear.

Recommendation Give descriptive names to all arguments and/or explain in a documentation comment.

```
69 function lzCompose(address _from, bytes32 _guid, bytes calldata
   ↪ _message, address, bytes calldata) external payable {
```

CVF-21 INFO

- **Category** Bad datatype
- **Source** StargateV2Receiver.sol

Recommendation The type for the "_from" argument should be "IPool".

Client Comment *Disagree.*

```
69 function lzCompose(address _from, bytes32 _guid, bytes calldata
   ↪ _message, address, bytes calldata) external payable {
```

CVF-22 INFO

- **Category** Bad datatype
- **Source** StargateV2Receiver.sol

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

Client Comment *I don't like specifying an interface if the address is not always expected to use it. In this case the address can be a placeholder for the native asset and so will not be using IERC20 functions*

```
97 function execute(address token, uint256 amount, bytes calldata data)
   ↪  public {
```

```
121 function _transfer(address token, address receiver, uint256 amount)
    ↪ internal {
```

CVF-23 INFO

- **Category** Bad datatype
- **Source** StargateV2Receiver.sol

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

Client Comment *Same as above.*

```
112 function sweep(address[] memory tokens) external onlyOwner {
```

CVF-24 FIXED

- **Category** Procedural
- **Source** StargateV2Receiver.sol

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

```
134 receive() external payable { }
```

CVF-25 FIXED

- **Category** Procedural
- **Source** EnsoRouter.sol

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

Recommendation Move the type definitions into the contract or move them into a separate file.

```
10 enum TokenType {
```

```
17 struct Token {
```

CVF-26 FIXED

- **Category** Procedural
- **Source** EnsoRouter.sol

Recommendation This contract should implement the “IRouter” interface.

```
22 contract EnsoRouter {
```

CVF-27 INFO

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

Recommendation The type for this variable should be “EnsoShortcuts”.

Client Comment *Disagree.*

```
25 address public immutable shortcuts;
```

CVF-28 INFO

- **Category** Procedural
- **Source** EnsoRouter.sol

Description These two functions are very similar.

Recommendation Refactor the code to avoid code duplication.

Client Comment *Disagree. In the case of the native asset we would be unnecessarily calling ‘abi.decode‘ in order to get the minAmountOut when all we are doing is checking the balance*

```
150 function _balance(Token calldata token, address receiver) internal
    ↪ view returns (uint256 balance) {
```

```
169 function _checkMinAmountOut(Token calldata token, address receiver,
    ↪ uint256 prevBalance) internal view {
```

CVF-29 INFO

- **Category** Procedural
- **Source** EnsoShortcuts.sol

Description We didn’t review this file.

```
4 import { AbstractEnsoShortcuts } from "./AbstractEnsoShortcuts.sol";
```

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