SPEARBIT

Optimism Upgrade Proposal #15a Review

Auditors

M4rio.eth, Lead Security Researcher

Milotruck, Lead Security Researcher

Report prepared by: Lucas Goiriz

May 1, 2025

Contents

1 About Spearbit

Spearbit is a decentralized network of expert security engineers offering reviews and other security related services to Web3 projects with the goal of creating a stronger ecosystem. Our network has experience on every part of the blockchain technology stack, including but not limited to protocol design, smart contracts and the Solidity compiler. Spearbit brings in untapped security talent by enabling expert freelance auditors seeking flexibility to work on interesting projects together.

Learn more about us at spearbit.com

2 Introduction

Optimism is a fast, stable, and scalable L2 blockchain built by Ethereum developers, for Ethereum developers. Built as a minimal extension to existing Ethereum software, Optimism's EVM-equivalent architecture scales your Ethereum apps without surprises. If it works on Ethereum, it works on Optimism at a fraction of the cost.

Disclaimer: This security review does not guarantee against a hack. It is a snapshot in time of "Upgrade Proposal 15a - Absolute Prestate Updates for Isthmus Activation Blob Preimage Fix - Optimism Mainnet and INK" according to the specific commit. Any modifications to the code will require a new security review.

3 Risk classification

Severity level	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

3.1 Impact

- High - leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
- Medium - global losses <10% or losses to only a subset of users, but still unacceptable.
- Low - losses will be annoying but bearable--applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.

3.2 Likelihood

- High - almost certain to happen, easy to perform, or not easy but highly incentivized
- Medium - only conditionally possible or incentivized, but still relatively likely
- Low - requires stars to align, or little-to-no incentive

3.3 Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

4 Executive Summary

Over the course of 4 days in total, OP Labs engaged with Spearbit to review Upgrade Proposal #15a.

Summary

Project Name	OP Labs
Repository	superchain-ops
Commit	1ee69747, ed149fe5
Type of Project	Infrastructure, L2
Audit Timeline	Apr 29th to May 3rd

5 Findings

5.1 Upgrade Proposal #15a - Absolute Prestate Updates for Isthmus Activation & Blob Preimage Fix - Optimism Mainnet and INK

Severity: Informational

5.1.1 Validation

This document can be used to validate the inputs and result of the execution of the upgrade transaction which you are signing. The steps are:

1. Validate the Domain and Message Hashes.
2. Verifying the state changes via the normalized state diff hash.
3. Verifying the transaction input.
4. Verifying the state changes.

5.1.2 Expected Domain and Message Hashes

First, we need to validate the domain and message hashes. These values should match both the values on your ledger an the values printed to the terminal when you run the task.

CAUTION:
Before signing, ensure the below hashes match what is on your ledger.
Optimism Foundation Upgrade Safe (`0x847B5c174615B1B7fDF770882256e2D3E95b9D92`**)**

- Domain Hash: `0xa4a9c312badf3fcaa05eafe5dc9bee8bd9316c78ee8b0bebe3115bb21b732672`
- Message Hash: `0xe742f60fe2e614478b475c5da80c7898f5e09668d158beb37d5131eeb34108f4`

Security Council (`0xc2819DC788505Aac350142A7A707BF9D03E3Bd03`**)**

- Domain Hash: `0xdf53d510b56e539b90b369ef08fce3631020fbf921e3136ea5f8747c20bce967`
- Message Hash: `0xe8dfdb92b25d01287028007b3c52a3a8b52a7204c6e8a2ebd7455ac8e7246a5f`

5.1.3 Normalized State Diff Hash Attestation:

The normalized state diff hash MUST match the hash created by the state changes attested to in the state diff audit report. As a signer, you are responsible for making sure this hash is correct. Please compare the hash below with the hash in the audit report.

Normalized hash: `0x4d50717185117827e3265c4183bfad6a0e839821a189342d38134f2e63a9c3b1`.

5.1.4 Understanding Task Calldata:

This document provides a detailed analysis of the final calldata executed on-chain. By reconstructing the calldata, we can confirm that the execution precisely implements the approved upgrade plan with no unexpected modifications or side effects.

The calldata provided in the governance proposal is:

```
0x82ad56cb000000000000000000000000000000000000000000000000000000000000002000000000000000000000000000000 ⌋
↪   000000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000020 ⌋
↪   0000000000000000000000003a1f523a4bc09cd344a2745a108bb0398288094f00000000000000000000000000000000000 ⌋
↪   000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000060000000 ⌋
↪   00000000000000000000000000000000000000000000000000000001049a72745b000000000000000000000000000000000 ⌋
↪   000000000000000000000000000002000000000000000000000000000000000000000000000000000000000000000020000 ⌋
↪   00000000000000000000229047fed2591dbec1ef1118d64f7af3db9eb290000000000000000000000000543ba4aadbab8f9 ⌋
↪   025686bd03993043599c6fb0403682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b60000000000 ⌋
↪   0000000000000062c0a111929fa32cec2f76adba54c16afb6e8364000000000000000000000000d56045e68956fce2576e6 ⌋
↪   80c95a4750cf8241f7903682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b60000000000000000 ⌋
↪   0000000000000000000000000000000000000000
```

5.1.5 Inputs to `Multicall3Delegatecall.aggregate3()`

The calldata from the governance proposal is the arguments to the `aggregate3()` function of the `Multicall3Delegatecall` contract, at `0x93dc480940585d9961bfceab58124ffd3d60f76a`.

The command to decode the calldata is:

```
cast decode-calldata "aggregate3((address,bool,bytes)[])" 0x82ad56cb00000000000000000000000000000000000 ⌋
↪   000000000000000000000000000200000000000000000000000000000000000000000000000000000000000000001000000 ⌋
↪   00000000000000000000000000000000000000000000000000000000200000000000000000000000003a1f523a4bc09cd34 ⌋
↪   4a2745a108bb0398288094f0000000000000000000000000000000000000000000000000000000000000000000000000000 ⌋
↪   000000000000000000000000000000000000000000000000006000000000000000000000000000000000000000000000000 ⌋
↪   000000000000001049a72745b000000000000000000000000000000000000000000000000000000000000002000000000000 ⌋
↪   000000000000000000000000000000000000000000000000000002000000000000000000000000229047fed2591dbec1ef1 ⌋
↪   118d64f7af3db9eb290000000000000000000000000543ba4aadbab8f9025686bd03993043599c6fb0403682932cec7ce0a ⌋
↪   3874b19675a6bbc923054a7b321efc7d3835187b172494b600000000000000000000000062c0a111929fa32cec2f76adba5 ⌋
↪   4c16afb6e8364000000000000000000000000d56045e68956fce2576e680c95a4750cf8241f7903682932cec7ce0a3874b1 ⌋
↪   9675a6bbc923054a7b321efc7d3835187b172494b600000000000000000000000000000000000000000000000000000000
```

The decoded arguments is an array with a single tuple of three elements:

```
[
    (
        0x3A1f523a4bc09cd344A2745a108Bb0398288094F,
        false
        0x9a72745b00000000000000000000000000000000000000000000000000000000000000200000000000000000000000 ⌋
        ↪   0000000000000000000000000000000000000000002000000000000000000000000229047fed2591dbec1ef1118 ⌋
        ↪   d64f7af3db9eb290000000000000000000000000543ba4aadbab8f9025686bd03993043599c6fb0403682932cec ⌋
        ↪   7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b600000000000000000000000062c0a111929fa3 ⌋
        ↪   2cec2f76adba54c16afb6e8364000000000000000000000000d56045e68956fce2576e680c95a4750cf8241f790 ⌋
        ↪   3682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b6
    )
]
```

This tuple is the `Call3` struct, which represents the parameters for a single delegatecall:

- `target`: The `OPContractsManager` contract.
- `allowFailure`: `false`.
- `calldata`: As shown above.

5.1.6 Inputs to `OPContractsManager.updatePrestate()`

The calldata in the `Call3` struct above is the arguments to the `updatePrestate()` function of the `OPContracts-Manager` contract, at `0x3A1f523a4bc09cd344A2745a108Bb0398288094F`.

The command to decode the calldata is:

```
cast decode-calldata "updatePrestate((address,address,bytes32)[])" 0x9a72745b00000000000000000000000000
↪  0000000000000000000000000000000000002000000000000000000000000000000000000000000000000000000000000
↪  002000000000000000000000000229047fed2591dbec1ef1118d64f7af3db9eb290000000000000000000000000543ba4aa
↪  dbab8f9025686bd03993043599c6fb0403682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b6000
↪  00000000000000000000062c0a111929fa32cec2f76adba54c16afb6e8364000000000000000000000000d56045e68956fc
↪  e2576e680c95a4750cf8241f7903682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b6
```

The decoded arguments is an array of tuples with three elements:

```
[
    (
        0x229047fed2591dbec1eF1118d64F7aF3dB9EB290,
        0x543bA4AADBAb8f9025686Bd03993043599c6fB04,
        0x03682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b6
    ),
    (
        0x62C0a111929fA32ceC2F76aDba54C16aFb6E8364,
        0xd56045E68956FCe2576E680c95a4750cf8241f79,
        0x03682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b6
    )
]
```

Each tuple is an `OpChainConfig` struct for each chain being updated:

1. OP Mainnet:

- `systemConfigProxy`: `0x229047fed2591dbec1eF1118d64F7aF3dB9EB290`.
- `proxyAdmin`: `0x543bA4AADBAb8f9025686Bd03993043599c6fB04`.
- `absolutePrestate`: `0x03682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b6`.

2. Ink Mainnet:

- `systemConfigProxy`: `0x62C0a111929fA32ceC2F76aDba54C16aFb6E8364`.
- `proxyAdmin`: `0xd56045E68956FCe2576E680c95a4750cf8241f79`.
- `absolutePrestate`: `0x03682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b6`.

As a result, `OPContractsManager.updatePrestate()` is called to update the prestate hash for both OP and Ink mainnet.

5.1.7 State Validation

For each contract listed in the state diff, please verify that no contracts or state changes shown in the Tenderly diff are missing from this document. Additionally, please verify that for each contract:

- The following state changes (and none others) are made to that contract. This validates that no unexpected state changes occur.
- All addresses (in section headers and storage values) match the provided name, using the Etherscan and Superchain Registry links provided. This validates the bytecode deployed at the addresses contains the correct logic.
- All key values match the semantic meaning provided, which can be validated using the storage layout links provided.

5.1.8 Generic Safe State Overrides

Note: The changes listed below do not include threshold, nonce and owner mapping overrides. These changes are listed and explained in the NESTED-VALIDATION.md file.

`0x10d7b35078d3baabb96dd45a9143b94be65b12cd` **(DisputeGameFactory) - Chain ID: 57073 (INK):**

- **Key:** `0x4d5a9bd2e41301728d41c8e705190becb4e74abe869f75bdb405b63716a35f9e`.
  - **Before:** `0x000000000000000000000000436bac2efe273e3f13eefeda2b3689c34591bca1`.
  - **After:** `0x00000000000000000000000040641a4023f0f4c66d7f8ade16497f4c947a7163`.
  - **Summary:** Replaces Dispute Game Implementation in the `DisputeGameFactory` contract.
  - **Detail:** This state update will replace the old Dispute Game implementation for the Game Type 1 (PERMISSIONED_CANNON) - 0x436bac2efe273e3f13eefeda2b3689c34591bca1 (old) ⇒ `0x40641a4023f0f4c66d7f8ade16497f4c947a7163` (new).
  - **Key Explanation:** The key represents the position in the gameImpls mapping of the targeted Game Type.

If we run the following command it will give us the exact position of Game Type 1 (PERMISSIONED_CANNON) `0x4d5a9bd2e41301728d41c8e705190becb4e74abe869f75bdb405b63716a35f9e`:

```
cast index uint32 1 101
```

- GameType is `uint32` type.
- Position in the mapping is `1`.
- Slot in the contract's storage is `101`.

Note: The new game implementation is identical to the old one, with the only update being the prestate set to `0x03682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b6`.

- **Key:** `0xffdfc1249c027f9191656349feb0761381bb32c9f557e01f419fd08754bf5a1b`.
  - **Before:** `0x000000000000000000000000499e30a3b1bdb03f554ffffae4c9c5edf31ca554`.
  - **After:** `0x0000000000000000000000003ccf7c31a3a8c1b8aaa9a18fc2d010dde4262342`.
  - **Summary:** Replaces Dispute Game Implementation in the `DisputeGameFactory` contract.
  - **Detail:** This state update will replace the old Dispute Game implementation for the Game Type 0 (CANNON) - 0x499e30a3b1bdb03f554ffffae4c9c5edf31ca554 (old) ⇒ `0x3ccf7c31a3a8c1b8aaa9a18fc2d010dde4262342` (new).
- **Key Explanation:** The key represents the position in the `gameImpls` of the targeted Game Type.

If we run the following command it will give us the exact position of Game Type 0 (CANNON) `0xffdfc1249c027f9191656349feb0761381bb32c9f557e01f419fd08754bf5a1b`:

```
cast index uint32 0 101
```

- GameType is `uint32` type.
- Position in the mapping is `0`.
- Slot in the contract's storage is `101`.

Note: The new game implementation is identical to the old one, with the only update being the prestate set to `0x03682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b6`.

`0xe5965ab5962edc7477c8520243a95517cd252fa9` **(DisputeGameFactory) - Chain ID: 10 (Optimism Mainnet):**

- **Key:** `0x4d5a9bd2e41301728d41c8e705190becb4e74abe869f75bdb405b63716a35f9e`.
  - **Before:** `0x0000000000000000000000001ae178ebfeecd51709432ea5f37845da0414edfe`.
  - **After:** `0x000000000000000000000000a1e0bacde89d899b3f24eef3d179cc335a24e777`.
  - **Summary:** Replaces Dispute Game Implementation in the `DisputeGameFactory` contract.

  - **Detail:** This state update will replace the old Dispute Game implementation for the Game Type 1 (PERMISSIONED_CANNON) - 0x1ae178ebfeecd51709432ea5f37845da0414edfe (old) ⇒ `0xa1e0bacde89d899b3f24eef3d179cc335a24e777` (new).

- **Key Explanation:** The key represents the position in the `gameImpls` of the targeted Game Type.

If we run the following command it will give us the exact position of Game Type 1 (PERMISSIONED_CANNON) `0x4d5a9bd2e41301728d41c8e705190becb4e74abe869f75bdb405b63716a35f9e`:

```
cast index uint32 1 101
```

- GameType is `uint32` type.
- Position in the mapping is `1`.
- Slot in the contract's storage is `101`.

Note: The new game implementation is identical to the old one, with the only update being the prestate set to `0x03682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b6`.

- **Key:** `0xffdfc1249c027f9191656349feb0761381bb32c9f557e01f419fd08754bf5a1b`.
  - **Before:** `0x0000000000000000000000005738a876359b48a65d35482c93b43e2c1147b32b`.
  - **After:** `0x00000000000000000000000089d68b1d63aaa0db4af1163e81f56b76934292f8`.
  - **Summary:** Replaces Dispute Game Implementation in the `DisputeGameFactory` contract.
  - **Detail:** This state update will replace the old Dispute Game implementation for the Game Type 0 (CANNON) - 0x5738a876359b48a65d35482c93b43e2c1147b32b (old) ⇒ `0x89d68b1d63aaa0db4af1163e81f56b76934292f8` (new).
- **Key Explanation:** The key represents the position in the `gameImpls` of the targeted Game Type.

If we run the following command it will give us the exact position of Game Type 0 (CANNON) `0xffdfc1249c027f9191656349feb0761381bb32c9f557e01f419fd08754bf5a1b`:

```
cast index uint32 0 101
```

- GameType is `uint32` type.
- Position in the mapping is `0`.
- Slot in the contract's storage is `101`.

Note: The new game implementation is identical to the old one, with the only update being the prestate set to `0x03682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b6`.

`0x24424336F04440b1c28685a38303aC33C9D14a25` **(LivenessGuard):**

> **IMPORTANT:**
> Security Council Only.

THIS STATE DIFF ONLY APPEARS WHEN SIGNING FOR THE COUNCIL AND DOES NOT NEED TO BE CHECKED BY SIGNERS.

The details are explained in NESTED-VALIDATION.md.

`0x5a0Aae59D09fccBdDb6C6CcEB07B7279367C3d2A` **(Superchain ProxyAdminOwner):**

- Nonce increments see below.
- `approvedHashes` mapping updates are explained in detail in NESTED-VALIDATION.md. The key computations are:

- **Foundation only**

  ```
  SAFE_SIGNER=0x847B5c174615B1B7fDF770882256e2D3E95b9D92
  SAFE_HASH=0x410dacd36755998923076d5c5f115b77116f3e479a9a5cecf45f6c2dab3da479
  cast index bytes32 $SAFE_HASH $(cast index address $SAFE_SIGNER 8)
  ```

  Key: `0xea44a27dff7f1fec743500257a14e44c424876595dfb8c1eaf765eecdd3c4f41`.

- **Security Council only**

  ```
  SAFE_SIGNER=0xc2819DC788505Aac350142A7A707BF9D03E3Bd03
  SAFE_HASH=0x410dacd36755998923076d5c5f115b77116f3e479a9a5cecf45f6c2dab3da479
  cast index bytes32 $SAFE_HASH $(cast index address $SAFE_SIGNER 8)
  ```

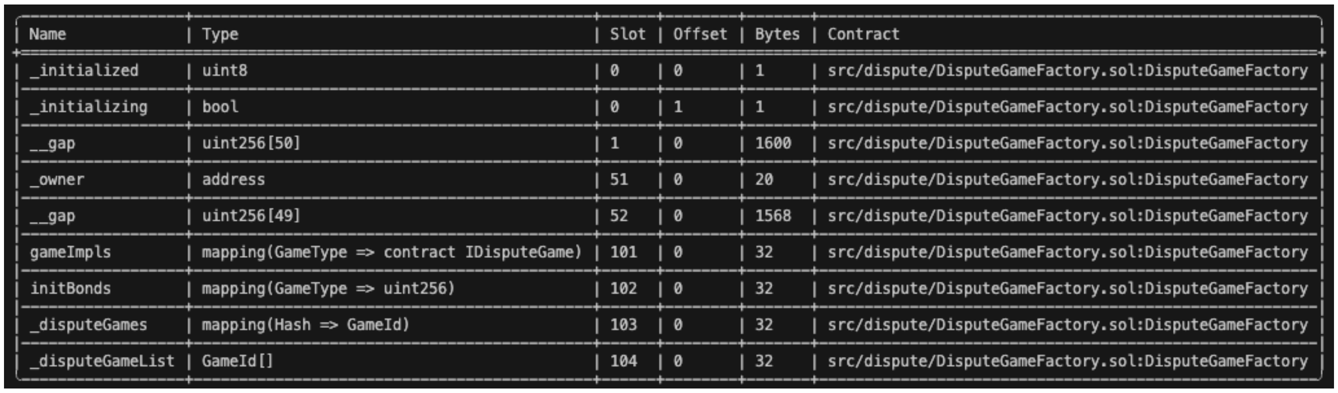
  Key: `0xb32ab0e2f892afb0356b7eb63cab3a3ba9ad4d3a01899d832360c55ddfa4a785`.

5.1.9 Nonce increments

- Contract deployments are shown as nonce increments from 0 to 1.
  - `0x3cCF7C31a3A8C1b8aaA9A18FC2d010dDE4262342` - Permissionless [CANON] GameType Implementation for Ink.
  - `0x40641A4023f0F4C66D7f8Ade16497f4C947A7163` - Permissioned [PERMISSIONED_CANNON] GameType Implementation for Ink.
  - `0x89D68b1D63AAA0db4af1163e81f56B76934292F8` - Permissionless [CANON] GameType Implementation for OP Mainnet.
  - `0xa1E0baCde89d899B3f24eEF3D179cC335A24E777` - Permissioned [PERMISSIONED_CANNON] GameType Implementation for OP Mainnet.
- The remaining nonce increments are for the Safes and EOAs that are involved in the simulation. The details are described in the generic NESTED-VALIDATION.md document.
  - `<sender-address>` - Sender address of the Tenderly transaction (Your ledger or first owner on the nested safe (if you're simulating)).
  - `0x5a0Aae59D09fccBdDb6C6CcEB07B7279367C3d2A` - Superchain ProxyAdminOwner.
    * Contract nonce `14` → `18` - four contract deployments above.
    * Safe nonce (slot `0x5`) `14` → `15`.
  - Only one of the following nonce increments, depending on which Owner Safe is simulated.
    * `0x847B5c174615B1B7fDF770882256e2D3E95b9D92` - Foundation Upgrade Safe `24` → `25`.
    * `0xc2819DC788505Aac350142A7A707BF9D03E3Bd03` - Security Council Safe `25` → `26`.

5.1.10 Supplementary Material

The following is the storage slots layout of the `DisputeGameFactory`.



```
+----------------+------------------------------------------+------+--------+-------+------------------------------------------------------+
| Name           | Type                                     | Slot | Offset | Bytes | Contract                                             |
+=========================================================================================================================================+
| _initialized   | uint8                                    | 0    | 0      | 1     | src/dispute/DisputeGameFactory.sol:DisputeGameFactory |
|----------------+------------------------------------------+------+--------+-------+------------------------------------------------------|
| _initializing  | bool                                     | 0    | 1      | 1     | src/dispute/DisputeGameFactory.sol:DisputeGameFactory |
|----------------+------------------------------------------+------+--------+-------+------------------------------------------------------|
| __gap          | uint256[50]                              | 1    | 0      | 1600  | src/dispute/DisputeGameFactory.sol:DisputeGameFactory |
|----------------+------------------------------------------+------+--------+-------+------------------------------------------------------|
| _owner         | address                                  | 51   | 0      | 20    | src/dispute/DisputeGameFactory.sol:DisputeGameFactory |
|----------------+------------------------------------------+------+--------+-------+------------------------------------------------------|
| __gap          | uint256[49]                              | 52   | 0      | 1568  | src/dispute/DisputeGameFactory.sol:DisputeGameFactory |
|----------------+------------------------------------------+------+--------+-------+------------------------------------------------------|
| gameImpls      | mapping(GameType => contract IDisputeGame) | 101  | 0      | 32    | src/dispute/DisputeGameFactory.sol:DisputeGameFactory |
|----------------+------------------------------------------+------+--------+-------+------------------------------------------------------|
| initBonds      | mapping(GameType => uint256)             | 102  | 0      | 32    | src/dispute/DisputeGameFactory.sol:DisputeGameFactory |
|----------------+------------------------------------------+------+--------+-------+------------------------------------------------------|
| _disputeGames  | mapping(Hash => GameId)                  | 103  | 0      | 32    | src/dispute/DisputeGameFactory.sol:DisputeGameFactory |
|----------------+------------------------------------------+------+--------+-------+------------------------------------------------------|
| _disputeGameList | GameId[]                               | 104  | 0      | 32    | src/dispute/DisputeGameFactory.sol:DisputeGameFactory |
+----------------+------------------------------------------+------+--------+-------+------------------------------------------------------+
```

5.2 Upgrade Proposal #15a - Absolute Prestate Updates for Isthmus Activation & Blob Preimage Fix - UNICHAIN

Severity: Informational

5.2.1 Validation

This document can be used to validate the inputs and result of the execution of the upgrade transaction which you are. signing. The steps are:

1. Validate the Domain and Message Hashes.
2. Verifying the state changes via the normalized state diff hash.
3. Verifying the transaction input.
4. Verifying the state changes.

5.2.2 Expected Domain and Message Hashes

First, we need to validate the domain and message hashes. These values should match both the values on your ledger and the values printed to the terminal when you run the task.

CAUTION:
Before signing, ensure the below hashes match what is on your ledger.
Unichain Upgrade Safe (Chain Governor) (`0xb0c4C487C5cf6d67807Bc2008c66fa7e2cE744EC`**)**
- Domain Hash: `0x4f0b6efb6c01fa7e127a0ff87beefbeb53e056d30d3216c5ac70371b909ca66d`
- Message Hash: `0x393727497cdd4c2a8f2a198643b44956ce007757d0400d6d977191318d06aea8`

Optimism Foundation Upgrade Safe (`0x847B5c174615B1B7fDF770882256e2D3E95b9D92`**)**
- Domain Hash: `0xa4a9c312badf3fcaa05eafe5dc9bee8bd9316c78ee8b0bebe3115bb21b732672`
- Message Hash: `0x5a5cc02357b2f7a6836b2921063b549f077410c3d423d972c0029512f400a3c3`

Security Council (`0xc2819DC788505Aac350142A7A707BF9D03E3Bd03`**)**
- Domain Hash: `0xdf53d510b56e539b90b369ef08fce3631020fbf921e3136ea5f8747c20bce967`
- Message Hash: `0xbfe796bd508232de1207a8668e26b13a3c4fdd8486b7b6a0636586bb045cb489`

5.2.3 Normalized State Diff Hash Attestation

The normalized state diff hash MUST match the hash created by the state changes attested to in the state diff audit report. As a signer, you are responsible for making sure this hash is correct. Please compare the hash below with the hash in the audit report.

Normalized hash: `0x5a3f19f595ad7baf0483c96aa23a6bfe7c74b64eb5333a069650017ae4faa790`.

5.2.4 Understanding Task Calldata

This document provides a detailed analysis of the final calldata executed on-chain. By reconstructing the calldata, we can confirm that the execution precisely implements the approved upgrade plan with no unexpected modifications or side effects.

The calldata provided in the governance proposal for Unichain is:

```
0x82ad56cb0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000 ⌟
↪   000000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000020 ⌟
↪   0000000000000000000000003a1f523a4bc09cd344a2745a108bb0398288094f00000000000000000000000000000000000 ⌟
↪   000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000060000000 ⌟
↪   000000000000000000000000000000000000000000000000000000000a49a72745b000000000000000000000000000000000 ⌟
↪   000000000000000000000000000002000000000000000000000000000000000000000000000000000000000000000010000 ⌟
↪   00000000000000000000c407398d063f942febbcc6f80a156b47f3f1bda60000000000000000000000003b73fa8d82f511a ⌟
↪   3cae17b5a26e4e1a2d5e2f2a403682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b60000000000 ⌟
↪   0000000000000000000000000000000000000000000000
```

5.2.5 Inputs to `Multicall3Delegatecall.aggregate3()`

The calldata from the governance proposal is the arguments to the `aggregate3()` function of the `Multicall3Delegatecall` contract, at `0x93dc480940585d9961bfceab58124ffd3d60f76a`.

The command to decode the calldata is:

```
cast decode-calldata "aggregate3((address,bool,bytes)[])"
↪   0x82ad56cb0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000 ⌟
↪   000000000000000000000000000000000000001000000000000000000000000000000000000000000000000000000000000 ⌟
↪   0020000000000000000000000003a1f523a4bc09cd344a2745a108bb0398288094f0000000000000000000000000000000 ⌟
↪   000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000006000 ⌟
↪   000000000000000000000000000000000000000000000000000000000000a49a72745b00000000000000000000000000000 ⌟
↪   000000000000000000000000000000000200000000000000000000000000000000000000000000000000000000000000001 ⌟
↪   000000000000000000000000c407398d063f942febbcc6f80a156b47f3f1bda60000000000000000000000003b73fa8d82f ⌟
↪   511a3cae17b5a26e4e1a2d5e2f2a403682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b6000000 ⌟
↪   00000000000000000000000000000000000000000000000000
```

The decoded arguments is an array with a single tuple of three elements:

```
[
    (
        0x3A1f523a4bc09cd344A2745a108Bb0398288094F,
        false,
        0x9a72745b0000000000000000000000000000000000000000000000000000000000000020000000000000000000000 ⌟
        ↪   0000000000000000000000000000000000000000010000000000000000000000000c407398d063f942febbcc6f8 ⌟
        ↪   0a156b47f3f1bda60000000000000000000000003b73fa8d82f511a3cae17b5a26e4e1a2d5e2f2a403682932cec ⌟
        ↪   7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b6
    )
]
```

This tuple is the `Call3` struct, which represents the parameters for a single delegatecall:

- `target`: The `OPContractsManager` contract.
- `allowFailure`: `false`.
- `calldata`: As shown above.

5.2.6 Inputs to `OPContractsManager.updatePrestate()`

The calldata in the `Call3` struct above is the arguments to the `updatePrestate()` function of the `OPContracts-Manager` contract, at `0x3A1f523a4bc09cd344A2745a108Bb0398288094F`.

The command to decode the calldata is:

```
cast decode-calldata "updatePrestate((address,address,bytes32)[])" 0x9a72745b00000000000000000000000000
↪   000000000000000000000000000000000000200000000000000000000000000000000000000000000000000000000000000
↪   001000000000000000000000000c407398d063f942febbcc6f80a156b47f3f1bda6000000000000000000000000003b73fa8d
↪   82f511a3cae17b5a26e4e1a2d5e2f2a403682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b6
```

The decoded arguments is an array with a single tuple of three elements:

```
[
    (
        0xc407398d063f942feBbcC6F80a156b47F3f1BDA6,
        0x3B73Fa8d82f511A3caE17B5a26E4E1a2d5E2f2A4,
        0x03682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b6
    )
]
```

This tuple is an `OpChainConfig` struct for the chain being updated, which is Unichain:

- `systemConfigProxy`: `0xc407398d063f942feBbcC6F80a156b47F3f1BDA6`.
- `proxyAdmin`: `0x543bA4AADBAb8f9025686Bd03993043599c6fB044`.
- `absolutePrestate`: `0x03682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b6`.

As a result, `OPContractsManager.updatePrestate()` is called to update the prestate hash for Unichain mainnet.

5.2.7 State Validation

For each contract listed in the state diff, please verify that no contracts or state changes shown in the Tenderly diff are missing from this document. Additionally, please verify that for each contract:

- The following state changes (and none others) are made to that contract. This validates that no unexpected state changes occur.
- All addresses (in section headers and storage values) match the provided name, using the Etherscan and Superchain Registry links provided. This validates the bytecode deployed at the addresses contains the correct logic.
- All key values match the semantic meaning provided, which can be validated using the storage layout links provided.

5.2.8 Generic Safe State Overrides

Note: The changes listed below do not include threshold, nonce and owner mapping overrides. These changes are listed and explained in the NESTED-VALIDATION.md file.

`0x2f12d621a16e2d3285929c9996f478508951dfe4` **(DisputeGameFactory) - Chain ID: 130 (UNICHAIN):**

- **Key:** `0x4d5a9bd2e41301728d41c8e705190becb4e74abe869f75bdb405b63716a35f9e`.
  - **Before:** `0x00000000000000000000000067d59ac1166ba17612be0edf275187e38cbf9b99`.
  - **After:** `0x000000000000000000000000485272c0703020e1354328a1aba3ca767997bed3`.
  - **Summary:** Replaces Dispute Game Implementation in the `DisputeGameFactory` contract.
  - **Detail:** This state update will replace the old Dispute Game implementation for the Game Type 1 (PERMISSIONED_CANNON) - 0x67d59ac1166ba17612be0edf275187e38cbf9b99 (old) ⇒ `0x485272c0703020e1354328a1aba3ca767997bed3` (new).
- **Key Explanation:** The key represents the position in the gameImpls mapping of the targeted Game Type.

If we run the following command it will give us the exact position of Game Type 1 (PERMISSIONED_CANNON) `0x4d5a9bd2e41301728d41c8e705190becb4e74abe869f75bdb405b63716a35f9e`:

```
cast index uint32 1 101
```

- GameType is `uint32` type.
- Position in the mapping is `1`.
- Slot in the contract's storage is `101`.

Note: The new game implementation is identical to the old one, with the only update being the prestate set to `0x03682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b6`.

- **Key:** `0xffdfc1249c027f9191656349feb0761381bb32c9f557e01f419fd08754bf5a1b`.
  - **Before:** `0x00000000000000000000000056ebb9eae4f33ceaed3672446e3812d77f8a8a2c`.
  - **After:** `0x00000000000000000000000057a3b42698dc1e4fb905c9ab970154e178296991`.
  - **Summary:** Replaces Dispute Game Implementation in the `DisputeGameFactory` contract.
  - **Detail:** This state update will replace the old Dispute Game implementation for the Game Type 0 (CANNON) - 0x56ebb9eae4f33ceaed3672446e3812d77f8a8a2c (old) ⇒ `0x57a3b42698dc1e4fb905c9ab970154e178296991` (new).
- **Key Explanation:** The key represents the position in the `gameImpls` of the targeted Game Type.

If we run the following command it will give us the exact position of Game Type 0 (CANNON) `0xffdfc1249c027f9191656349feb0761381bb32c9f557e01f419fd08754bf5a1b`:

```
cast index uint32 0 101
```

- GameType is `uint32` type.
- Position in the mapping is `0`.
- Slot in the contract's storage is `101`.

Note: The new game implementation is identical to the old one, with the only update being the prestate set to `0x03682932cec7ce0a3874b19675a6bbc923054a7b321efc7d3835187b172494b6`.

0x9343c452dec3251fe99D9Fd29b74c5b9CD1751a6 (LivenessGuard Unichain):

> **IMPORTANT:**
> Unichain Safe Only

THIS STATE DIFF ONLY APPEARS WHEN SIGNING FOR THE UNICHAIN SAFE AND DOES NOT NEED TO BE CHECKED BY SIGNERS.

The details are explained in NESTED-VALIDATION.md.

0x24424336F04440b1c28685a38303aC33C9D14a25 (LivenessGuard Security Council):

> **IMPORTANT:**
> Security Council Only.

THIS STATE DIFF ONLY APPEARS WHEN SIGNING FOR THE COUNCIL AND DOES NOT NEED TO BE CHECKED BY SIGNERS.

The details are explained in NESTED-VALIDATION.md.

`0x6d5b183f538abb8572f5cd17109c617b994d5833` **(Unichain ProxyAdminOwner):**

- Nonce increments see below.
- `approvedHashes` mapping updates are explained in detail in NESTED-VALIDATION.md. The key computations are:
  - **Unichain Safe only**.

    ```
    SAFE_SIGNER=0xb0c4C487C5cf6d67807Bc2008c66fa7e2cE744EC
    SAFE_HASH=0x1ddd958de5bc75389847abb6cd0d8551f0ecfdaf763b9c80e935dbb1c37a3948
    cast index bytes32 $SAFE_HASH $(cast index address $SAFE_SIGNER 8)
    ```

    Key: `0xf8504c099de345eb1c403a30d49833b4834f40d609b6b2107b81927e309b987a`.
  - **Optimism Foundation only**.

    ```
    SAFE_SIGNER=0x847B5c174615B1B7fDF770882256e2D3E95b9D92
    SAFE_HASH=0x1ddd958de5bc75389847abb6cd0d8551f0ecfdaf763b9c80e935dbb1c37a3948
    cast index bytes32 $SAFE_HASH $(cast index address $SAFE_SIGNER 8)
    ```

    Key: `0xab2f364801a9ab669e9ddf4ec9b8d06c52acca51c9626e5242dd8a9b79a1f0aa`.
  - **Security Council only**.

    ```
    SAFE_SIGNER=0xc2819DC788505Aac350142A7A707BF9D03E3Bd03
    SAFE_HASH=0x1ddd958de5bc75389847abb6cd0d8551f0ecfdaf763b9c80e935dbb1c37a3948
    cast index bytes32 $SAFE_HASH $(cast index address $SAFE_SIGNER 8)
    ```

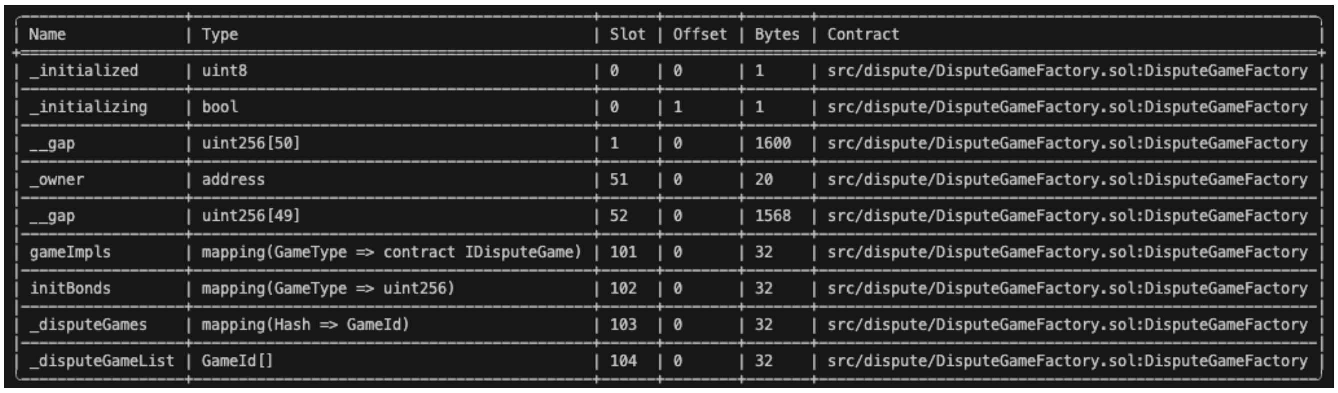
    Key: `0x488861e7a26dcec539aebd39e2015ecbaaa7c5924c668939a8cfe1af67718786`.

5.2.9 Nonce increments

- Contract deployments are shown as nonce increments from 0 to 1.
  - `0x485272c0703020e1354328A1aBa3ca767997BEd3` - Permissioned [PERMISSIONED_CANNON] GameType Implementation for Unichain Mainnet.
  - `0x57a3B42698DC1e4Fb905c9ab970154e178296991` - Permissionless [CANON] GameType Implementation for Unichain Mainnet.
- The remaining nonce increments are for the Safes and EOAs that are involved in the simulation. The details are described in the generic NESTED-VALIDATION.md document.
  - `<sender-address>` - Sender address of the Tenderly transaction (Your ledger or first owner on the nested safe (if you're simulating)).
  - `0x6d5B183F538ABB8572F5cD17109c617b994D5833` - Unichain ProxyAdminOwner.
    * Contract nonce `6` → `8` - two contract deployments above.
    * Safe nonce (slot `0x5`) `4` → `5`.
- Only one of the following nonce increments, depending on which Owner Safe is simulated.
  - `0xb0c4C487C5cf6d67807Bc2008c66fa7e2cE744EC` - Unichain Operations Safe `10` → `11`.
  - `0x847B5c174615B1B7fDF770882256e2D3E95b9D92` - Foundation Upgrade Safe `25` → `26`.
  - `0xc2819DC788505Aac350142A7A707BF9D03E3Bd03` - Security Council Safe `26` → `27`.

5.2.10 Supplementary Material

The following is the storage slots layout of the `DisputeGameFactory` contract:



```
╭─────────────────┬──────────────────────────────────────────┬──────┬────────┬───────┬─────────────────────────────────────────────────────╮
| Name            | Type                                     | Slot | Offset | Bytes | Contract                                            |
+==========================================================================================================================================+
| _initialized    | uint8                                    | 0    | 0      | 1     | src/dispute/DisputeGameFactory.sol:DisputeGameFactory |
|-----------------+------------------------------------------+------+--------+-------+-----------------------------------------------------|
| _initializing   | bool                                     | 0    | 1      | 1     | src/dispute/DisputeGameFactory.sol:DisputeGameFactory |
|-----------------+------------------------------------------+------+--------+-------+-----------------------------------------------------|
| __gap           | uint256[50]                              | 1    | 0      | 1600  | src/dispute/DisputeGameFactory.sol:DisputeGameFactory |
|-----------------+------------------------------------------+------+--------+-------+-----------------------------------------------------|
| _owner          | address                                  | 51   | 0      | 20    | src/dispute/DisputeGameFactory.sol:DisputeGameFactory |
|-----------------+------------------------------------------+------+--------+-------+-----------------------------------------------------|
| __gap           | uint256[49]                              | 52   | 0      | 1568  | src/dispute/DisputeGameFactory.sol:DisputeGameFactory |
|-----------------+------------------------------------------+------+--------+-------+-----------------------------------------------------|
| gameImpls       | mapping(GameType => contract IDisputeGame) | 101  | 0      | 32    | src/dispute/DisputeGameFactory.sol:DisputeGameFactory |
|-----------------+------------------------------------------+------+--------+-------+-----------------------------------------------------|
| initBonds       | mapping(GameType => uint256)             | 102  | 0      | 32    | src/dispute/DisputeGameFactory.sol:DisputeGameFactory |
|-----------------+------------------------------------------+------+--------+-------+-----------------------------------------------------|
| _disputeGames   | mapping(Hash => GameId)                  | 103  | 0      | 32    | src/dispute/DisputeGameFactory.sol:DisputeGameFactory |
|-----------------+------------------------------------------+------+--------+-------+-----------------------------------------------------|
| _disputeGameList | GameId[]                                | 104  | 0      | 32    | src/dispute/DisputeGameFactory.sol:DisputeGameFactory |
╰─────────────────┴──────────────────────────────────────────┴──────┴────────┴───────┴─────────────────────────────────────────────────────╯
```