L2 Sequencer Health Flag

The idea behind an Optimistic Rollup (OR) type of protocol is to move all execution off-chain and keep all transaction data available on-chain. Such protocols have a special off-chain component, a Sequencer, that executes and rolls up the Layer 2 transactions by batching multiple transactions into a single one.

If a sequencer becomes unavailable, it becomes impossible to access read/write APIs that consumers are using so every dapp will be down for 95% of the users, except those that know how to interact with the Layer 1 OR contracts. In this case, it would be unfair to continue providing service on your dApp, as only 5 % of the users can use it. Note, this doesn't mean that the Layer 2 network has stopped, as OR is not an actual chain.

Overview

The L2 Sequencer Health Flag helps mitigate potential exploits when the Sequencer is unavailable by notifying the corresponding OR protocol to raise a flag on Layer 2.

The L2 Sequencer Health Flag consists of three actors:

1. Chainlink Cluster (a group of validator nodes) - executes the OCR Job every heartbeat "T" (the minimum frequency the Chainlink feed is configured to be updated)
2. The actual OCR feed reporting the Sequencer status - could be used for external users on Layer 1 to check OR protocol (e.g. Arbitrum) status
3. Validator - gets triggered by the OCR feed and executes the raise or lower flag action if the current answer is different from the previous one

Checking the Sequencer Status

If you have contracts that rely on Layer 2 Chainlink Data Feeds, you should add an extra check for each of your contracts. To implement, use the following sample:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";
import "@chainlink/contracts/src/v0.8/interfaces/FlagsInterface.sol";

contract ArbitrumPriceConsumer {
    // Identifier of the Sequencer offline flag on the Flags contract
    address constant private FLAG_ARBITRUM_SEQ_OFFLINE = address(bytes20(bytes
    AggregatorV3Interface internal priceFeed;
    FlagsInterface internal chainlinkFlags;

    /**
     * Network: Arbitrum Rinkeby
     * Aggregator: ETH/USD
```

```
     * Agg Address: 0x5f0423B1a6935dc5596e7A24d98532b67A0AeFd8
     * Flags Address: 0x491B1dDA0A8fa069bbC1125133A975BF4e85a91b
     */
    constructor() {
        priceFeed = AggregatorV3Interface(0x5f0423B1a6935dc5596e7A24d98532b67A
        chainlinkFlags = FlagsInterface(0x491B1dDA0A8fa069bbC1125133A975BF4e85
    }

    /**
     * Returns the latest price
     */
    function getThePrice() public view returns (int) {
        bool isRaised = chainlinkFlags.getFlag(FLAG_ARBITRUM_SEQ_OFFLINE);
        if (isRaised) {
                // If flag is raised we shouldn't perform any critical operati
            revert("Chainlink feeds are not being updated");
        }
        (
            uint80 roundID,
            int price,
            uint startedAt,
            uint timeStamp,
            uint80 answeredInRound
        ) = priceFeed.latestRoundData();
        return price;
    }
}
```

> **Important**
>
> Flag should be checked using `address(bytes20(bytes32(uint256(keccak256("chainlink.flags.arbitrum-seq-offline")) - 1)))` which translates into `0xa438451D6458044c3c8CD2f6f31c91ac882A6d91`

A raised flag will determine that the feed wasn't updated in "T" time and its data can be considered stale. In other words, the Sequencer went offline and your contract shouldn't perform any critical operations. When the Sequencer comes back up again and the Layer 2 Chainlink Data Feeds are updated, you can continue using your contracts as usual.

Contract Addresses

> **Note**
>
> These contract addresses are on L2, and should therefore only be read from L2.

Mainnet Contracts

Name	Address
Arbitrum Mainnet Flags Contract	0×3C14e07Edd0dC67442FA96f1Ec6999c57E810a83

Rinkeby Contracts

Name	Address
Arbitrum Rinkeby Flags Contract	0×491B1dDA0A8fa069bbC1125133A975BF4e85a91b

> **Note**
>
> Healthcheck Proxy Feed returns `1` when the Sequencer is offline and `0` when Sequencer is available