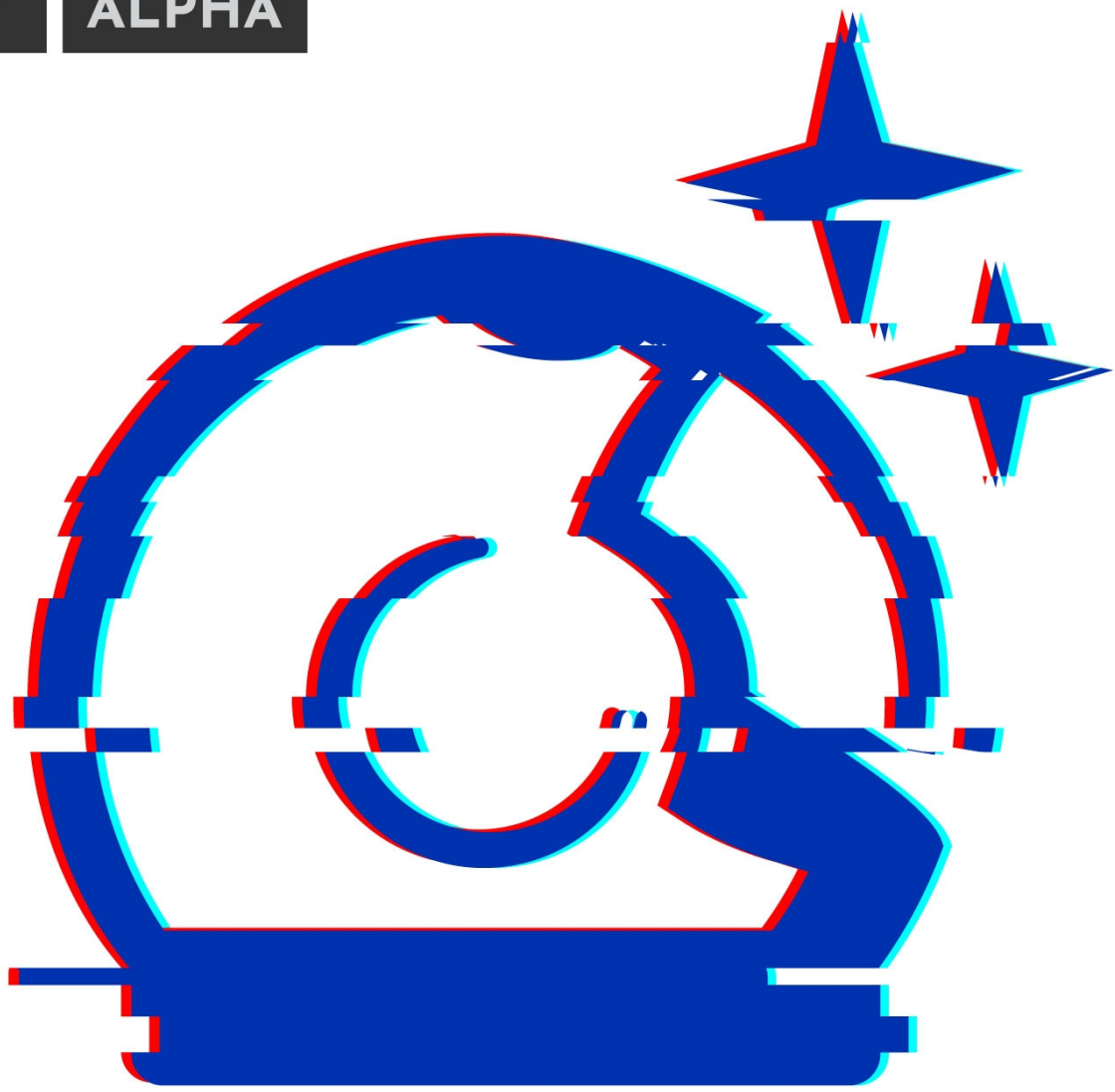
MONO CEROS ALPHA



PROJECT

Hop Exchange Smart Contracts

CLIENT

Hop Protocol

DATE

April 2021

REVIEWERS

Daniel Luca

@cleanunicorn

Table of Contents

Details

- **Client** Hop Protocol
- **Date** April 2021
- **Lead reviewer** Daniel Luca (@cleanunicorn)
- **Reviewers** Daniel Luca (@cleanunicorn)
- **Repository**: Hop Exchange Smart Contracts

- **Commit hash** `f486cc2f1f5086f1fd7a3bba8645bc2b0fd700c2`
- **Technologies**
  - Solidity
  - Node.JS

Issues Summary

SEVERITY	OPEN	CLOSED
Informational	3	0
Minor	8	0
Medium	1	0
Major	0	0

Executive summary

This report represents the results of the engagement with **Hop Protocol** to review **Hop Exchange Smart Contracts**.

The review was conducted over the course of **2.5 weeks** from **March 29 to April 14, 2021**. A total of **13 person-days** were spent reviewing the code.

Week 1

During the first week, we started by viewing the kick-off call and getting more familiar with the architecture and the overall setup.

In the second day of the audit, we updated the commit hash to include a Merkle tree update. The `MerkleUtils` library was swapped out with the one Optimism is using. The scope was increased to include a few other contracts that are part of the system and extend the original scope included contracts.

During the 3rd day, after getting more familiar with the code and the overall architecture, we proceeded to read the whitepaper and the rest of the documentation. Our goal was to understand all the key terms and components to make sure the definitions are completely understood in the context of the system.

At the end of the week, we had another meeting with the development team to go together through a few scenarios. The important scenarios were related to entry points and data flow when a user crosses one of the bridges, i.e., user crossing L1 -> L2, user crossing L2a -> L2b (different layer 2 chains).

Week 2

During the second week we set a daily meeting with the development team up until the end of the review to discuss new findings and ask any outstanding questions.

On Monday we identified a few small issues related to the Automated Market Maker Wrapper (AmmWrapper) and add a few questions for the development team. We proceeded to ask these questions in our daily sync. Some of the raised issues were already identified by the development team and fixed in their internal audit.

On Tuesday the development team shared with us an updated document with a few scenarios and their entry points that we should review.

Text version of shared interaction doc

Rendered version of the interaction below:

Sending and Bonding

Bonder Online

L1 → L2

1. User calls `L1_Bridge.sendToL2()` (Transfer will show up automatically on layer 2)

L2 → L1

1. User calls `L2_AMM_Wapper.swapAndSend()` (Uniswap parameters are both 0)
2. Bonder calls `L1_Bridge.bondWithdrawal()` (Transfer is now complete)

L2 → L2

1. User calls `L2_AMM_Wapper.swapAndSend()` (Uniswap parameters are set)
2. Bonder calls `L2_DestinationBridge.bondWithdrawalAndDistribute()` (Transfer is now complete)

Bonder Offline

L1 → L2

1. User calls `L1_Bridge.sendToL2()` (Transfer will show up automatically on layer 2)
    - Note: There is no difference here if the bonder is online or offline — the bonder plays no role in this transfer.

L2 → L1 & L2 → L2

1. User calls `L2_Bridge.swapAndSend()`
2. Bonder is offline so the `bond...()` functions are never called and `l2_bridge.commitTransfers()` is called after 100 transactions are sent or 4 hours
3. Canonical withdrawal time (~7 days for Optimism) goes by
4. `L2_Bridge.commitTransfers()` call happens and "atomically" calls `L1_Bridge.confirmTransferRoot()` which "atomically" calls `DestinationBridge.setTransferRoot()` (this bridge can be on either L1 or L2)
5. User (or relayer) calls `DestinationBridge.withdraw()` (this bridge can be on either L1 or L2)

Committing and Settling

Bonder Online

Commit and settle TransferRoot

1. Bonder calls `L2_Bridge.commitTransfers()`
2. Bonder calls `L1_Bridge.bondTransferRoot()`
3. Anyone (but probably the Bonder) calls `settleBondedWithdrawals()` on every destination bridge for all transfers transfer
    - Note: `settleBondedWithdrawals()` should be called most of the time, however, it is possible to also settle withdrawals one at a time with `settleBondedWithdrawal()`

Bonder Offline

Commit and settle TransferRoot

- There is nothing to be done here if the bonder is online. The user will send a transaction, wait for the transaction to be committed by the canonical messenger (~7 days for Optimism), and then call `DestinationBridge.withdraw()` (this bridge can be on either L1 or L2).

On Wednesday we continued to draft issues based on the feedback received and continued to do the manual review.

At the end of the week we agreed to add another 3 person days in order to fully cover the review.

Week 3

We continued the review, mostly focusing on the challenge, confirmation and root id confirmation.

At the end of the review period we finalized the report and presented it to the development team.

Scope

The initial review focused on the Hop Exchange Smart Contracts identified by the commit hash `f486cc2f1f5086f1fd7a3bba8645bc2b0fd700c2`.

We focused on manually reviewing the codebase, searching for security issues such as, but not limited to re-entrancy problems, transaction ordering, block timestamp dependency, exception handling, call stack depth limitation, integer overflow/underflow, self-destructible contracts, unsecured balance, use of origin, gas costly patterns, architectural problems, code readability.

Includes:

- ./bridges/Accounting.sol
- ./bridges/Bridge.sol
- ./bridges/L1_Bridge.sol
- ./bridges/L1_ETH_Bridge.sol
- ./bridges/L1_ERC20_Bridge.sol
- ./bridges/L2_Bridge.sol
- ./bridges/L2_AmmWrapper.sol
- ./wrappers/ArbitrumMessengerWrapper.sol
- ./wrappers/MessengerWrapper.sol
- ./wrappers/OptimismMessengerWrapper.sol
- ./wrappers/XDaiMessengerWrapper.sol

Excludes:

All other contracts

Recommendations

We identified a few possible general improvements that are not security issues during the review, which will bring value to the developers and the community reviewing and using the product.

Improve tests

Currently the tests do not pass. This is because the optimism compiler npm package was changed since the code was frozen, up until the tests were run on our end. This isn't a problem with the code itself, but pinning the npm packages to a fixed version will prevent similar situations.

A sample `.env.example` config file should also be provided to anyone that wants to run the tests.

Improve the testing process to make sure all functionality is first tested and then implemented, basically writing the tests first and the implementation after. This way ensures each line of code is tested and no line of code is superfluous.

Set up Continuous Integration

Use one of the platforms that offer Continuous Integration services and implement a list of actions that compile, test, run coverage and create alerts when the pipeline fails.

Because the repository is hosted on GitHub, the most painless way to set up the Continuous Integration is through GitHub Actions.

Setting up the workflow can start based on this example template.

```
name: Continuous Integration

on:
  push:
    branches: [master]
  pull_request:
    branches: [master]

jobs:
  build:
    name: Build and test
    runs-on: ubuntu-latest
    strategy:
      matrix:
        node-version: [12.x]
    steps:
    - uses: actions/checkout@v2
    - name: Use Node.js $
      uses: actions/setup-node@v1
      with:
```

```
        node-version: $
    - run: npm ci
    - run: cp ./config.sample.js ./config.js
    - run: npm test

  coverage:
    name: Coverage
    needs: build
    runs-on: ubuntu-latest
    strategy:
      matrix:
        node-version: [12.x]
    steps:
    - uses: actions/checkout@v2
    - name: Use Node.js $
      uses: actions/setup-node@v1
      with:
        node-version: $
    - run: npm ci
    - run: cp ./config.sample.js ./config.js
    - run: npm run coverage
    - uses: actions/upload-artifact@v2
      with:
        name: Coverage $
        path: |
          coverage/
```

This CI template activates on pushes and pull requests on the **master** branch.

```
on:
  push:
    branches: [master]
  pull_request:
    branches: [master]
```

It uses an Ubuntu Docker image as a base for setting up the project.

```
    runs-on: ubuntu-latest
```

Multiple Node.js versions can be used to check integration. However, because this is not primarily a Node.js project, multiple versions don't provide added value.

```
    strategy:
      matrix:
        node-version: [12.x]
```

A script item should be added in the `scripts` section of package.json that runs all tests.

```
{
   "script": {
      "test": "buidler test"
   }
}
```

This can then be called by running `npm test` after setting up the dependencies with `npm ci`.

If any hidden variables need to be defined, you can set them up in a local version of `./config.sample.js` (locally named `./config.js`). If you decide to do that, you should also add `./config.js` in `.gitignore` to make sure no hidden variables are pushed to the public repository. The sample config file `./config.sample.js` should be sufficient to pass the test suite.

```
    steps:
    - uses: actions/checkout@v2
    - name: Use Node.js $
      uses: actions/setup-node@v1
      with:
        node-version: $
    - run: npm ci
    - run: cp ./config.sample.js ./config.js
    - run: npm test
```

You can also choose to run coverage and upload the generated artifacts.

```
    - run: npm run coverage
    - uses: actions/upload-artifact@v2
      with:
        name: Coverage $
        path: |
          coverage/
```

At the moment, checking the artifacts is not that easy, because one needs to download the zip archive, unpack it and check it. However, the coverage can be checked in the Actions section once it's set up.

Issues

`L2_Bridge._distribute` might mint tokens for the Layer 1 Bridge and they'll be locked

Status Open Severity Medium

Description

The method `distribute` can only be called by the Layer 1 Bridge.

code/contracts/bridges/L2_Bridge.sol#L176-L184

```
    function distribute(
        address recipient,
        uint256 amount,
        uint256 amountOutMin,
        uint256 deadline,
        uint256 relayerFee
    )
        external
        onlyL1Bridge
```

The `external` method calls the `internal` method `_distribute` in order to distribute tokens.

code/contracts/bridges/L2_Bridge.sol#L263

```
    function _distribute(address recipient, uint256 amount, uint256 amountOutMin, uint256 deadli
```

If the `fee` is positive, it will mint some tokens for the caller.

code/contracts/bridges/L2_Bridge.sol#L264-L266

```
        if (fee > 0) {
            hToken.mint(msg.sender, fee);
        }
```

In this case, the caller is the Layer 1 Bridge as stated above.

However, the L1 Bridge doesn't seem to have a method to move the tokens.

```
+  L1_Bridge (Bridge)
   - [Pub] <Constructor> #
      - modifiers: Bridge
   - [Ext] sendToL2 ($)
   - [Ext] bondTransferRoot #
      - modifiers: onlyBonder,requirePositiveBalance
   - [Ext] confirmTransferRoot #
      - modifiers: onlyL2Bridge
   - [Int] _distributeTransferRoot #
   - [Ext] challengeTransferBond ($)
   - [Ext] resolveChallenge #
   - [Int] _additionalDebit
   - [Int] _requireIsGovernance #
   - [Ext] setGovernance #
      - modifiers: onlyGovernance
```

```
- [Ext] setCrossDomainMessengerWrapper #
  - modifiers: onlyGovernance
- [Ext] setChainIdDepositsPaused #
  - modifiers: onlyGovernance
- [Ext] setChallengeAmountMultiplier #
  - modifiers: onlyGovernance
- [Ext] setChallengeAmountDivisor #
  - modifiers: onlyGovernance
- [Ext] setChallengePeriodAndTimeSlotSize #
  - modifiers: onlyGovernance
- [Ext] setChallengeResolutionPeriod #
  - modifiers: onlyGovernance
- [Ext] setMinTransferRootBondDelay #
  - modifiers: onlyGovernance
- [Pub] getBondForTransferAmount
- [Pub] getChallengeAmountForTransferAmount
- [Pub] getTimeSlot
```

In which case the tokens that represent the fee, will remain locked in the Layer 1 Bridge contract.

Recommendation

Consider adding an additional argument to point to the receiver of the fee and pass that along through the internal `_distribute` method. This is especially useful for the `bondWithdrawalAndDistribute` method.

The fee doesn't seem necessary if the Layer 1 Bridge calls the external `distribute` method because the amount is not bonded.

Consider emitting events when updating the L2_Bridge config

Status Open Severity Minor

Description

All of these methods update important configurations for the bridge.

code/contracts/bridges/L2_Bridge.sol#L292-L337

```
/* ========== External Config Management Functions ========== */

function setAmmWrapper(L2_AmmWrapper _ammWrapper) external onlyGovernance {
    ammWrapper = _ammWrapper;
}

function setL1BridgeAddress(address _l1BridgeAddress) external onlyGovernance {
    l1BridgeAddress = _l1BridgeAddress;
```

```
    }

    function setL1MessengerWrapperAddress(address _l1MessengerWrapperAddress) external onlyGover
        l1MessengerWrapperAddress = _l1MessengerWrapperAddress;
    }

    function setMessengerGasLimit(uint256 _messengerGasLimit) external onlyGovernance {
        messengerGasLimit = _messengerGasLimit;
    }

    function addSupportedChainIds(uint256[] calldata chainIds) external onlyGovernance {
        for (uint256 i = 0; i < chainIds.length; i++) {
            supportedChainIds[chainIds[i]] = true;
        }
    }

    function removeSupportedChainIds(uint256[] calldata chainIds) external onlyGovernance {
        for (uint256 i = 0; i < chainIds.length; i++) {
            supportedChainIds[chainIds[i]] = false;
        }
    }

    function setMinimumForceCommitDelay(uint256 _minimumForceCommitDelay) external onlyGovernanc
        minimumForceCommitDelay = _minimumForceCommitDelay;
    }

    function setMaxPendingTransfers(uint256 _maxPendingTransfers) external onlyGovernance {
        maxPendingTransfers = _maxPendingTransfers;
    }

    function setHopBridgeTokenOwner(address newOwner) external onlyGovernance {
        hToken.transferOwnership(newOwner);
    }

    function setMinimumBonderFeeRequirements(uint256 _minBonderBps, uint256 _minBonderFeeAbsolut
        minBonderBps = _minBonderBps;
        minBonderFeeAbsolute = _minBonderFeeAbsolute;
    }
```

Recommendation

Consider emitting events with the new values (and possibly old values) when these changes go through.

Similarly can be done for `L1_Bridge`.

Make `L2_Bridge.send` external

Status Open Severity Minor

Description

The method is only called externally. To signal it will not be called internally by the contract or by another implementation that extends the `L2_Bridge` it could be defined as `external`.

Recommendation

Consider defining the method as `external` if it's not called internally.

Committing transfers should fail early if the `destinationChainId` isn't supported

Status Open Severity Minor

Description

Anyone can call the method `commitTransfers` to commit the current pending transfers.

code/contracts/bridges/L2_Bridge.sol#L153-L160

```
    /**
     * @dev Aggregates all pending Transfers to the `destinationChainId` and sends them to the
     * L1_Bridge as a TransferRoot.
     * @param destinationChainId The chainId of the TransferRoot's destination chain
     */
    function commitTransfers(uint256 destinationChainId) external {
        uint256 minForceCommitTime = lastCommitTimeForChainId[destinationChainId].add(minimumFor
        require(minForceCommitTime < block.timestamp || getIsBonder(msg.sender), "L2_BRG: Only B
```

The Bonder can call this method anytime they want.

code/contracts/bridges/L2_Bridge.sol#L160

```
        require(minForceCommitTime < block.timestamp || getIsBonder(msg.sender), "L2_BRG: Only B
```

But any other actor can call the method if enough time has passed since the previous commit time.

If they specify an incorrect (unsupported) chain id, the method will still try to call `_commitTransfers`. In which case it should fail if no transfers are pending. It is better to fail earlier if the chain id is unsupported.

Recommendation

Check the provided `destinationChainId` to be valid.

```
require(supportedChainIds[chainId], "L2_BRG: destinationChainId is not supported");
```

In case the chain was previously supported and it is disabled right now, some transfers can remain in a pending state, still uncommitted. It is better to be more explicit about each state you want to support and how.

In case there are pending transfers, you should still allow committing the transfers, even if the chain is not supported anymore, but not allow creating new pending transfers.

Consider allowing pending transfers if the minimum time has passed, even if the chain id is not supported anymore, only if there are pending transfers. If the logic seems messy or too complicated to be handled in `commitTransfers`, consider creating a `flushTransfers` method just for this use case.

`L2_Bridge.setMinimumBonderFeeRequirements` should check `minBonderBps` for validity

Status Open Severity Minor

Description

The governance can set a minimum fee, as well as a minimum percentage by calling `setMinimumBonderFeeRequirements`.

code/contracts/bridges/L2_Bridge.sol#L334-L337

```
    function setMinimumBonderFeeRequirements(uint256 _minBonderBps, uint256 _minBonderFeeAbsolut
        minBonderBps = _minBonderBps;
        minBonderFeeAbsolute = _minBonderFeeAbsolute;
    }
```

The values could be checked for sanity, especially the percentage.

Recommendation

Check the value of `minBonderBps` to be something less than `10000`.

Improve gas usage in `L2_AmmWrapper`

Status Open Severity Minor

Description

The contract `L2_AmmWrapper` has a few storage variables.

code/contracts/bridges/L2_AmmWrapper.sol#L13-L17

```
    L2_Bridge public bridge;
    IERC20 public l2CanonicalToken;
    bool public l2CanonicalTokenIsEth;
    IERC20 public hToken;
    Swap public exchangeAddress;
```

These storage vars are set at deploy time.

code/contracts/bridges/L2_AmmWrapper.sol#L19-L34

```
    /// @notice When l2CanonicalTokenIsEth is true, l2CanonicalToken should be set to the WETH a
    constructor(
        L2_Bridge _bridge,
        IERC20 _l2CanonicalToken,
        bool _l2CanonicalTokenIsEth,
        IERC20 _hToken,
        Swap _exchangeAddress
    )
        public
    {
        bridge = _bridge;
        l2CanonicalToken = _l2CanonicalToken;
        l2CanonicalTokenIsEth = _l2CanonicalTokenIsEth;
        hToken = _hToken;


        exchangeAddress = _exchangeAddress;
    }
```

None of them are changed after the initial set.

This is why they could be defined as `immutable`. Solidity does not reserve a storage slot for immutables, but every occurrence is replaced in the bytecode by the respective value. This greatly reduces gas costs.

To have a better understanding of the gas cost reduction, two contracts were created.

The first contract defines a storage variable and sets it at deploy time. It also set the variable to `public`, this way Solidity will create a getter for it. This getter is important to measure the gas cost to access it.

```
contract A {
    bool public isSet;

    constructor(bool _isSet) public {
        isSet = _isSet;
    }
}
```

The second contract uses an `immutable` instead of a storage slot. That is the only difference between the two.

```
contract B {
    bool public immutable isSet;

    constructor(bool _isSet) public {
        isSet = _isSet;
    }
}
```

We've created the following table to show the differences in gas costs between the two approaches.

	deploy gas cost	getter gas cost
storage slot	47801	988
immutable	32403	182

Both examples were tested in Remix IDE, Solidity version 0.6.12, optimization runs 200.

Recommendation

Change all state variables to `immutable` to greatly reduce gas costs.

Similarly, change the state variables to `immutable` in `L2_Bridge`.

code/contracts/bridges/L2_Bridge.sol#L78-L80

```
        l1Governance = _l1Governance;
        hToken = _hToken;
        l2CanonicalToken = _l2CanonicalToken;
```

But also in these cases (not a complete list):

code/contracts/wrappers/ArbitrumMessengerWrapper.sol#L36-L39

```
        arbInbox = _arbInbox;
        arbBridge = arbInbox.bridge();
        defaultGasPrice = _defaultGasPrice;
        defaultCallValue = _defaultCallValue;
```

code/contracts/wrappers/OptimismMessengerWrapper.sol#L29

```
        l1MessengerAddress = _l1MessengerAddress;
```

code/contracts/wrappers/XDaiMessengerWrapper.sol#L35-L36

```
        l2ChainId = bytes32(_l2ChainId);
        ambBridge = _ambBridge;
```

There are instances where some state variables are never changed in other contracts. If you think you won't change them in the future, consider checking all contract constructors.

References

- Constant and Immutable State Variables

Events exist but they're not emitted in `Accounting`

Status Open Severity Minor

Description

The Accounting contract takes care of low-level accounting and has a couple of `external` methods that handle staking and unstaking of funds.

code/contracts/bridges/Accounting.sol#L118-L127

```
    /**
     * @dev Allows the bonder to deposit tokens and increase its credit balance
     * @param bonder The address being staked on
     * @param amount The amount being staked
     */
    function stake(address bonder, uint256 amount) external payable {
        require(_isBonder[bonder] == true, "ACT: Address is not bonder");
        _transferToBridge(msg.sender, amount);
        _addCredit(bonder, amount);
    }
```

code/contracts/bridges/Accounting.sol#L129-L136

```
    /**
     * @dev Allows the caller to withdraw any available balance and add to their debit balance
     * @param amount The amount being staked
     */
    function unstake(uint256 amount) external requirePositiveBalance {
        _addDebit(msg.sender, amount);
        _transferFromBridge(msg.sender, amount);
    }
```

A couple of matching events exist in the contract.

code/contracts/bridges/Accounting.sol#L28-L30

```
    event Stake (
        uint256 amount
    );
```

code/contracts/bridges/Accounting.sol#L32-L34

```
    event Unstake (
        uint256 amount
    );
```

But they are not emitted in this contract or in any contract that inherits `Accounting`.

Recommendation

Emit events when staking and unstaking.

Optimize `_ceilLog2` by using `uint256`

Status Open Severity Minor

Description

The method `Lib_MerkleTree._ceilLog2` is used in the `getMerkleRoot` method to calculate the total siblings count.

code/contracts/libraries/Lib_MerkleTree.sol#L144-L147

```
        require(
            _siblings.length == _ceilLog2(_totalLeaves),
            "Lib_MerkleTree: Total siblings does not correctly correspond to total leaves."
        );
```

As stated in the source code, it was copied from the Solidity examples repository.

code/contracts/libraries/Lib_MerkleTree.sol#L202-L203

```
        // Find the highest set bit (will be floor(log_2)).
        // Borrowed with <3 from https://github.com/ethereum/solidity-examples
```

The Solidity example repository includes this method as an example for other developers to learn how to use the language. Because it does not strive to be the optimal implementation, its purpose is to show different features of the language itself.

src/bits/Bits.sol#L87-L99

```
    // Computes the index of the highest bit set in 'self'.
    // Returns the highest bit set as an 'uint8'.
    // Requires that 'self != 0'.
```

```
    function highestBitSet(uint self) internal pure returns (uint8 highest) {
        require(self != 0);
        uint val = self;
        for (uint8 i = 128; i >= 1; i >>= 1) {
            if (val & (ONE << i) - 1 << i != 0) {
                highest += i;
                val >>= i;
            }
        }
    }
```

This is why some of the examples in the Solidity repository might not be gas optimized.

Using `uint8` in the loop forces the compiler to create sub-optimal code.

The following Solidity contract example was used to measure the difference between using `uint8` and `uint256` with a few example inputs.

```
contract FindBit {
    function find_highest_set_bit(
        uint value
    )
        public
        pure
        returns (
            uint
        )
    {
        uint _in = value;

        if (_in == 1) {
            return 0;
        }

        // Find the highest set bit (will be floor(log_2)).
        // Borrowed with <3 from https://github.com/ethereum/solidity-examples
        uint256 val = _in;
        uint256 highest = 0;
        for (uint8 i = 128; i >= 1; i >>= 1) {
            if (val & (uint(1) << i) - 1 << i != 0) {
                highest += i;
                val >>= i;
            }
        }

        // Increment by one if this is not a perfect logarithm.
        if ((uint(1) << highest) != _in) {
            highest += 1;
        }

        return highest;
```

```
    }
}
```

Changing the line from `uint8` to `uint256`

```
        for (uint8 i = 128; i >= 1; i >>= 1) {
```

```
        for (uint256 i = 128; i >= 1; i >>= 1) {
```

Gives the following different gas costs, while returning the same, identical results.

Solidity version 0.8.3+commit.8d00100c and 200 optimization runs were used in Remix IDE for tests.

input	output	uint8 gas	uint256 gas	difference
8	3	1876	1692	184
15	4	1951	1767	184
100	7	1951	1767	184
128	7	1980	1779	201
4294967295 (`0xffffffff`)	32	2263	2028	235
1099511627775 (`0xffffffffff`)	40	2159	1941	218
18446744073709552000 (`0xffffffffffffffff`)	64	2367	2115	252

Recommendation

Modify the loop iterator from `uint8` to `uint256` to reduce gas costs, while having the same output.

References

- Solidity example, highest set bit
- Optimism MerkleTree implementation

Emit events when adding and removing bonders

Status Open Severity Minor

Description

The methods `addBonder` and `removeBonder` respectively add and remove bonders.

code/contracts/bridges/Accounting.sol#L138-L145

```
    /**
     * @dev Add Bonder to allowlist
     * @param bonder The address being added as a Bonder
     */
    function addBonder(address bonder) external onlyGovernance {
        require(_isBonder[bonder] == false, "ACT: Address is already bonder");
        _isBonder[bonder] = true;
    }
```

code/contracts/bridges/Accounting.sol#L147-L154

```
    /**
     * @dev Remove Bonder from allowlist
     * @param bonder The address being removed as a Bonder
     */
    function removeBonder(address bonder) external onlyGovernance {
        require(_isBonder[bonder] == true, "ACT: Address is not bonder");
        _isBonder[bonder] = false;
    }
```

Also, the constructor adds a list of bonders to the bonder list.

code/contracts/bridges/Accounting.sol#L54-L59

```
    /// @dev Sets the bonder addresses
    constructor(address[] memory bonders) public {
        for (uint256 i = 0; i < bonders.length; i++) {
            _isBonder[bonders[i]] = true;
        }
    }
```

It might be beneficial to emit events when adding and removing bonders.

Recommendation

Consider emitting events when adding or removing bonders.

The method `getChainId` can be restricted to `pure` in Solidity <0.8.x

Status Open Severity Informational

Description

This method is set as `view` but can be restricted to `pure` as long as the Solidity version is `<0.8.x`.

code/contracts/bridges/Bridge.sol#L112

```
    function getChainId() public virtual view returns (uint256 chainId) {
```

Also, the *hack* to silence the state mutability can also be removed.

code/contracts/bridges/Bridge.sol#L113

```
        this; // Silence state mutability warning without generating any additional byte code
```

If you need the `view` mutability in order to allow other layer 2 implementations to first save and then retrieve the value from the contract state, keep the `view` modifier, otherwise consider switching to `pure`. Solidity does not do too many things differently but uses `STATICCALL` when calling the method to make sure the state is not changed.

Recommendation

Unless there's a reason to keep the `view`, consider changing to `pure`.

Some error messages in `L2_AmmWrapper` refer to `L2_Bridge`

Status Open Severity Informational

Description

There are a couple of error messages that refer to the L2 Bridge

code/contracts/bridges/L2_AmmWrapper.sol#L50

```
        require(amount >= bonderFee, "L2_BRG: Bonder fee cannot exceed amount");
```

code/contracts/bridges/L2_AmmWrapper.sol#L53

```
            require(msg.value == amount, "L2_BRG: Value does not match amount");
```

Typically the `L2_BRG` error prefix is found in `Bridge` and `L2_Bridge`.

Also, the other error messages seem to refer to the Uniswap Wrapper.

code/contracts/bridges/L2_AmmWrapper.sol#L56

```
            require(l2CanonicalToken.transferFrom(msg.sender, address(this), amount), "L2_UW: Tr
```

code/contracts/bridges/L2_AmmWrapper.sol#L59

```
        require(l2CanonicalToken.approve(address(exchangeAddress), amount), "L2_UW: Approve fail
```

code/contracts/bridges/L2_AmmWrapper.sol#L72-L73

```
        require(hToken.transferFrom(msg.sender, address(this), amount), "L2_UW: TransferFrom fai
        require(hToken.approve(address(exchangeAddress), amount), "L2_UW: Approve failed");
```

code/contracts/bridges/L2_AmmWrapper.sol#L88

```
            require(hToken.transfer(recipient, amount), "L2_UW: Transfer failed");
```

code/contracts/bridges/L2_AmmWrapper.sol#L95

```
            require(success, 'L2_UW: ETH transfer failed');
```

code/contracts/bridges/L2_AmmWrapper.sol#L97

```
            require(l2CanonicalToken.transfer(recipient, amountOut), "L2_UW: Transfer failed");
```

Recommendation

Update the error messages to correctly reflect the current contract.

Update comments to be more swapper agnostic

Status Open Severity Informational

Description

Uniswap is mentioned, make sure to replace with something like AMM.

code/contracts/bridges/L2_AmmWrapper.sol#L56

```
            require(l2CanonicalToken.transferFrom(msg.sender, address(this), amount), "L2_UW: Tr
```

code/contracts/bridges/L2_AmmWrapper.sol#L59

```
        require(l2CanonicalToken.approve(address(exchangeAddress), amount), "L2_UW: Approve fail
```

code/contracts/bridges/L2_AmmWrapper.sol#L72-L73

```
        require(hToken.transferFrom(msg.sender, address(this), amount), "L2_UW: TransferFrom fai
        require(hToken.approve(address(exchangeAddress), amount), "L2_UW: Approve failed");
```

code/contracts/bridges/L2_AmmWrapper.sol#L88

```
            require(hToken.transfer(recipient, amount), "L2_UW: Transfer failed");
```

code/contracts/bridges/L2_AmmWrapper.sol#L95

```
            require(success, 'L2_UW: ETH transfer failed');
```

code/contracts/bridges/L2_AmmWrapper.sol#L97

```
            require(l2CanonicalToken.transfer(recipient, amountOut), "L2_UW: Transfer failed");
```

Recommendation

Make sure to update the error messages to reflect the current contract.

Artifacts

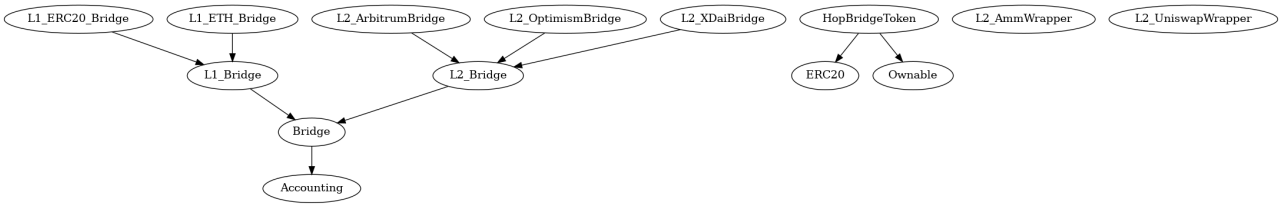
Surya

Sūrya is a utility tool for smart contract systems. It provides a number of visual outputs and information about the structure of smart contracts. It also supports querying the function call graph in multiple ways to aid in the manual inspection and control flow analysis of contracts.

Graphs

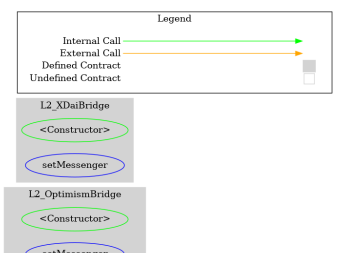
Inheritance

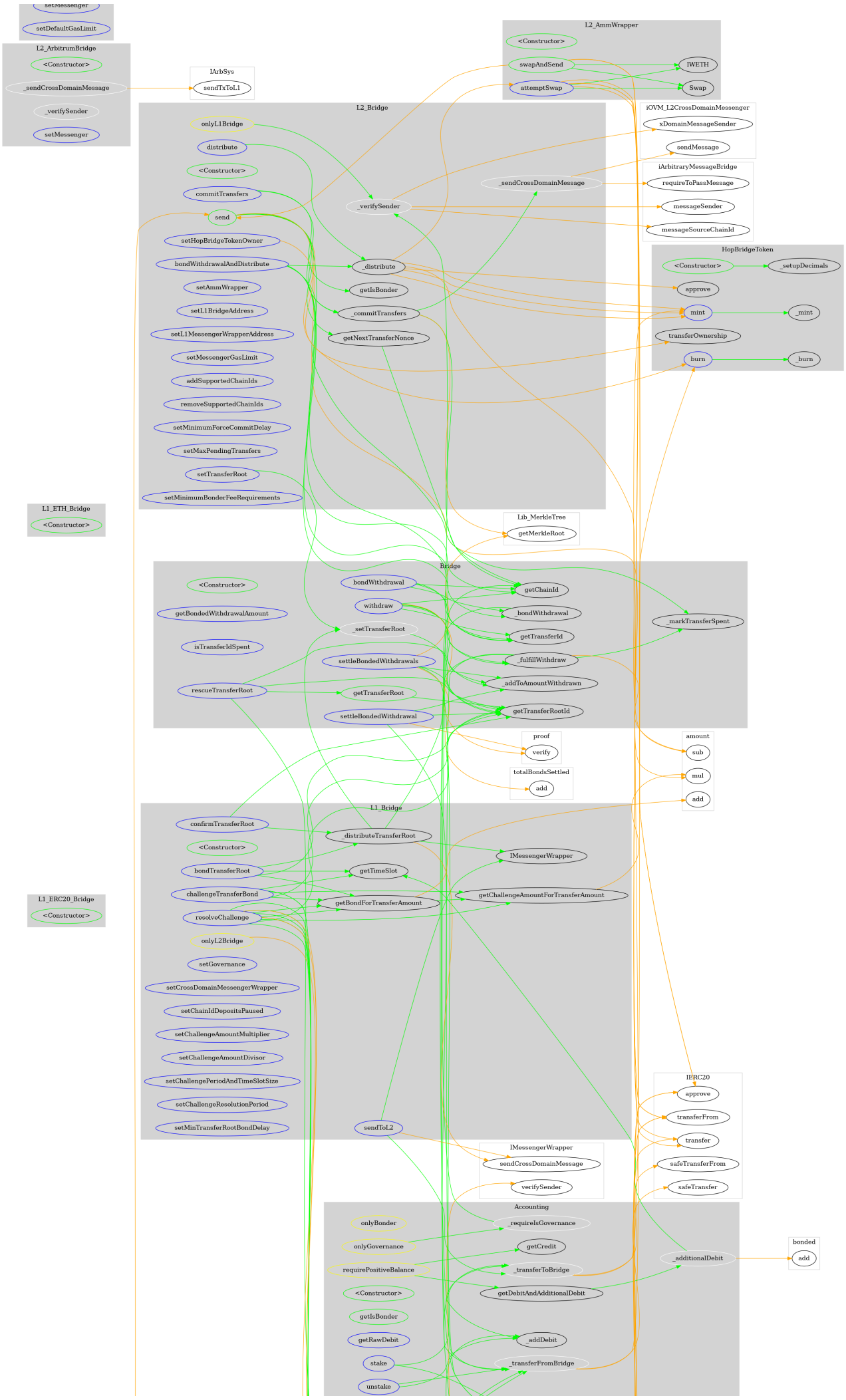
Display of the full inheritance path.

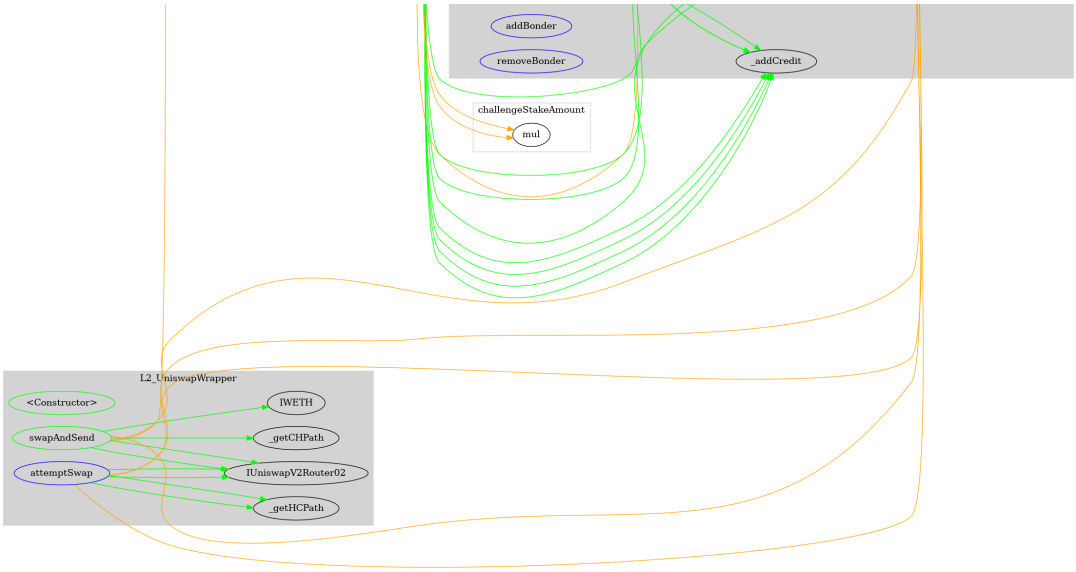


Full graph of interaction

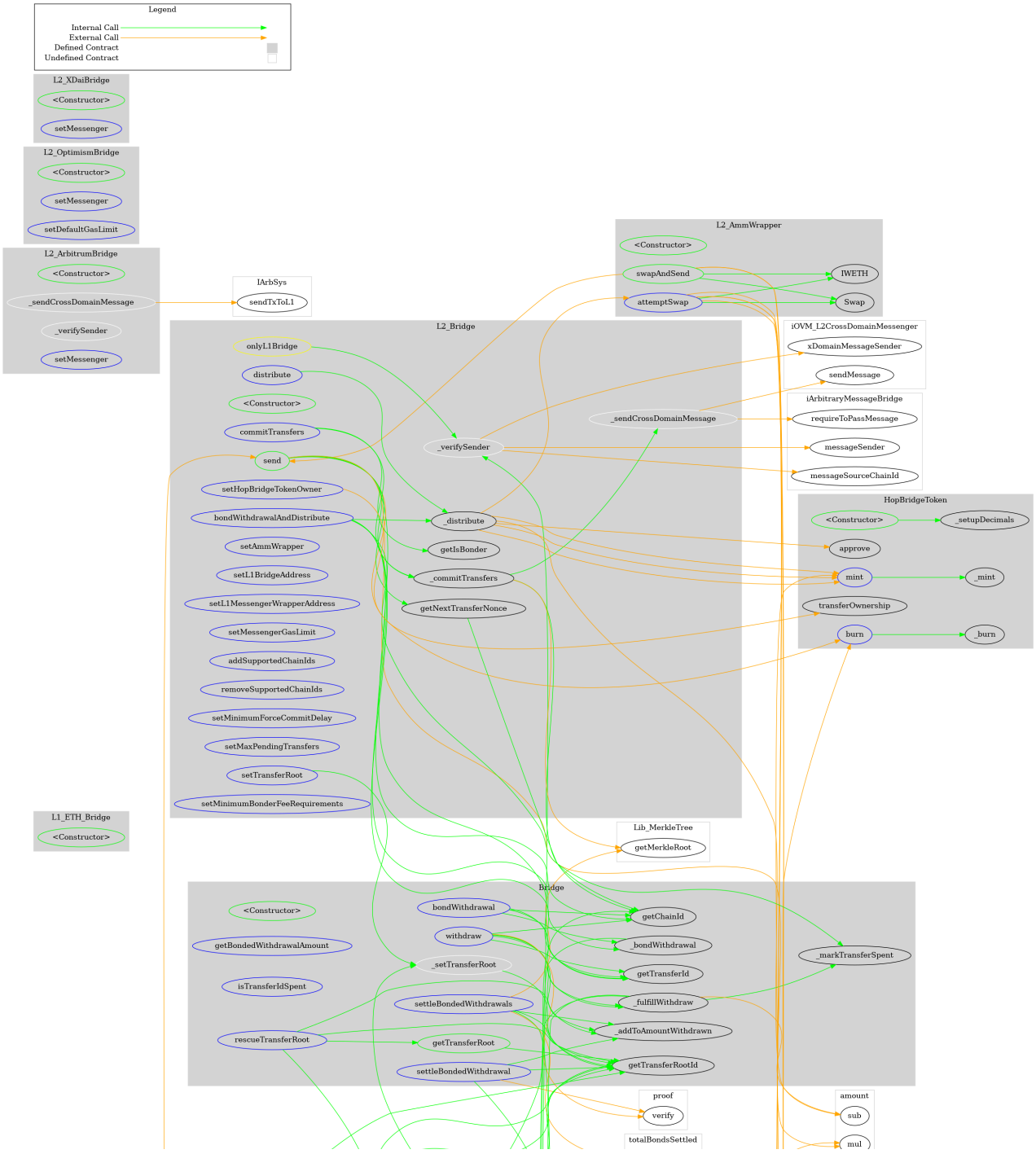
Some of the interactions are not visible in this graph because they are sent over the bridge by abi encoding and seding them over the messenger.

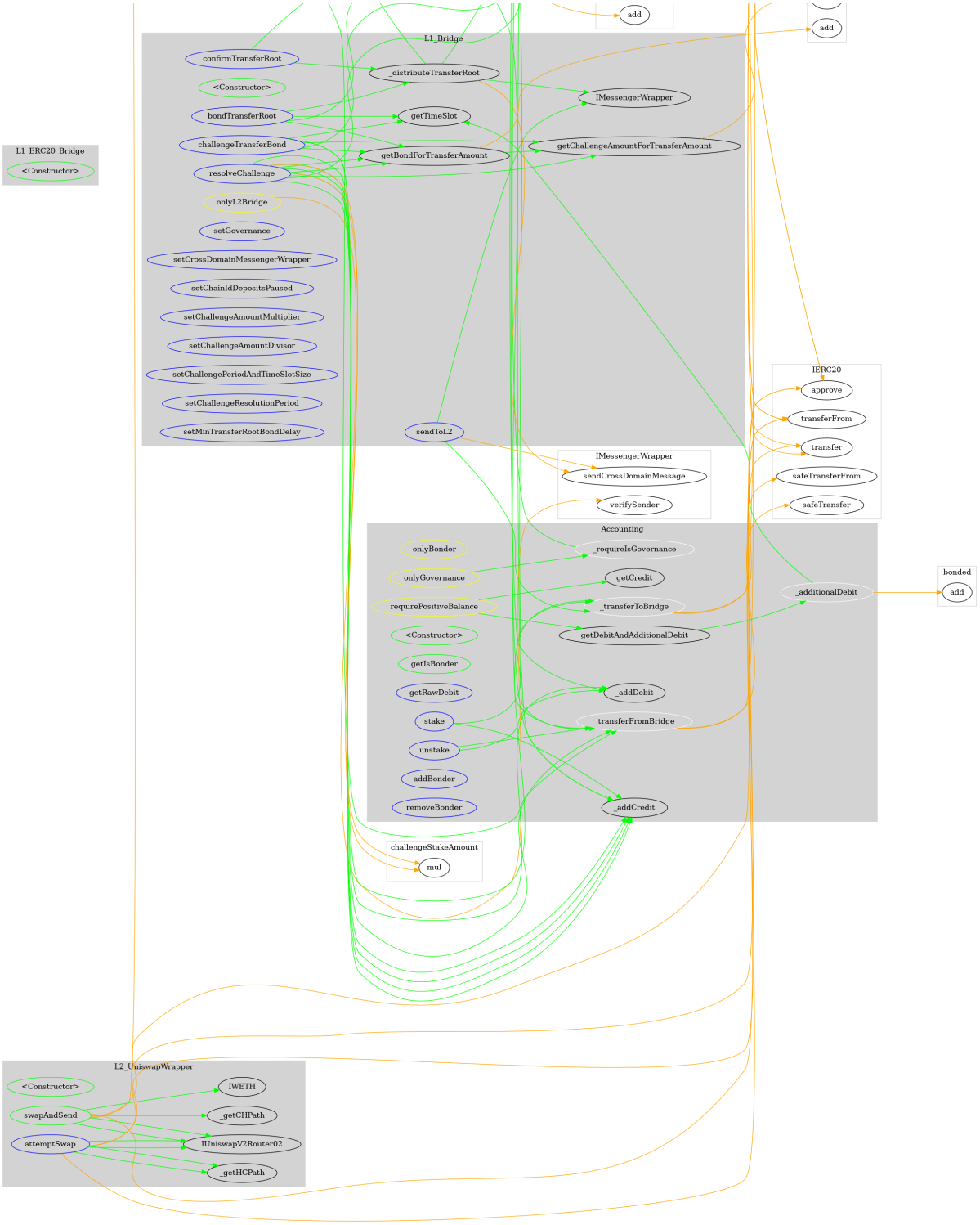



setMessenger
setDefaultGasLimit
L2_ArbitrumBridge
<Constructor>
_sendCrossDomainMessage
_verifySender
setMessenger
IArbSys
sendTxToL1
L2_AmmWrapper
<Constructor>
swapAndSend
attemptSwap
IWETH
Swap
L2_Bridge
onlyL1Bridge
distribute
<Constructor>
commitTransfers
send
setHopBridgeTokenOwner
bondWithdrawalAndDistribute
setAmmWrapper
setL1BridgeAddress
setL1MessengerWrapperAddress
setMessengerGasLimit
addSupportedChainIds
removeSupportedChainIds
setMinimumForceCommitDelay
setMaxPendingTransfers
setTransferRoot
setMinimumBonderFeeRequirements
_verifySender
_distribute
getIsBonder
_commitTransfers
getNextTransferNonce
_sendCrossDomainMessage
iOVM_L2CrossDomainMessenger
xDomainMessageSender
sendMessage
iArbitraryMessageBridge
requireToPassMessage
messageSender
messageSourceChainId
HopBridgeToken
<Constructor>
_setupDecimals
approve
mint
_mint
transferOwnership
burn
_burn
L1_ETH_Bridge
<Constructor>
Lib_MerkleTree
getMerkleRoot
Bridge
<Constructor>
getBondedWithdrawalAmount
isTransferIdSpent
rescueTransferRoot
bondWithdrawal
withdraw
setTransferRoot
settleBondedWithdrawals
getTransferRoot
settleBondedWithdrawal
getChainId
_bondWithdrawal
getTransferId
_fulfillWithdraw
_addToAmountWithdrawn
getTransferRootId
_markTransferSpent
proof
verify
totalBondsSettled
add
amount
sub
mul
add
L1_Bridge
confirmTransferRoot
<Constructor>
bondTransferRoot
challengeTransferBond
resolveChallenge
onlyL2Bridge
setGovernance
setCrossDomainMessengerWrapper
setChainIdDepositsPaused
setChallengeAmountMultiplier
setChallengeAmountDivisor
setChallengePeriodAndTimeSlotSize
setChallengeResolutionPeriod
setMinTransferRootBondDelay
_distributeTransferRoot
getTimeSlot
getBondForTransferAmount
sendToL2
IMessengerWrapper
getChallengeAmountForTransferAmount
L1_ERC20_Bridge
<Constructor>
IERC20
approve
transferFrom
transfer
safeTransferFrom
safeTransfer
IMessengerWrapper
sendCrossDomainMessage
verifySender
Accounting
onlyBonder
onlyGovernance
requirePositiveBalance
<Constructor>
getIsBonder
getRawDebit
stake
unstake
_requireIsGovernance
getCredit
_transferToBridge
getDebitAndAdditionalDebit
_addDebit
_transferFromBridge
_additionalDebit
bonded
add

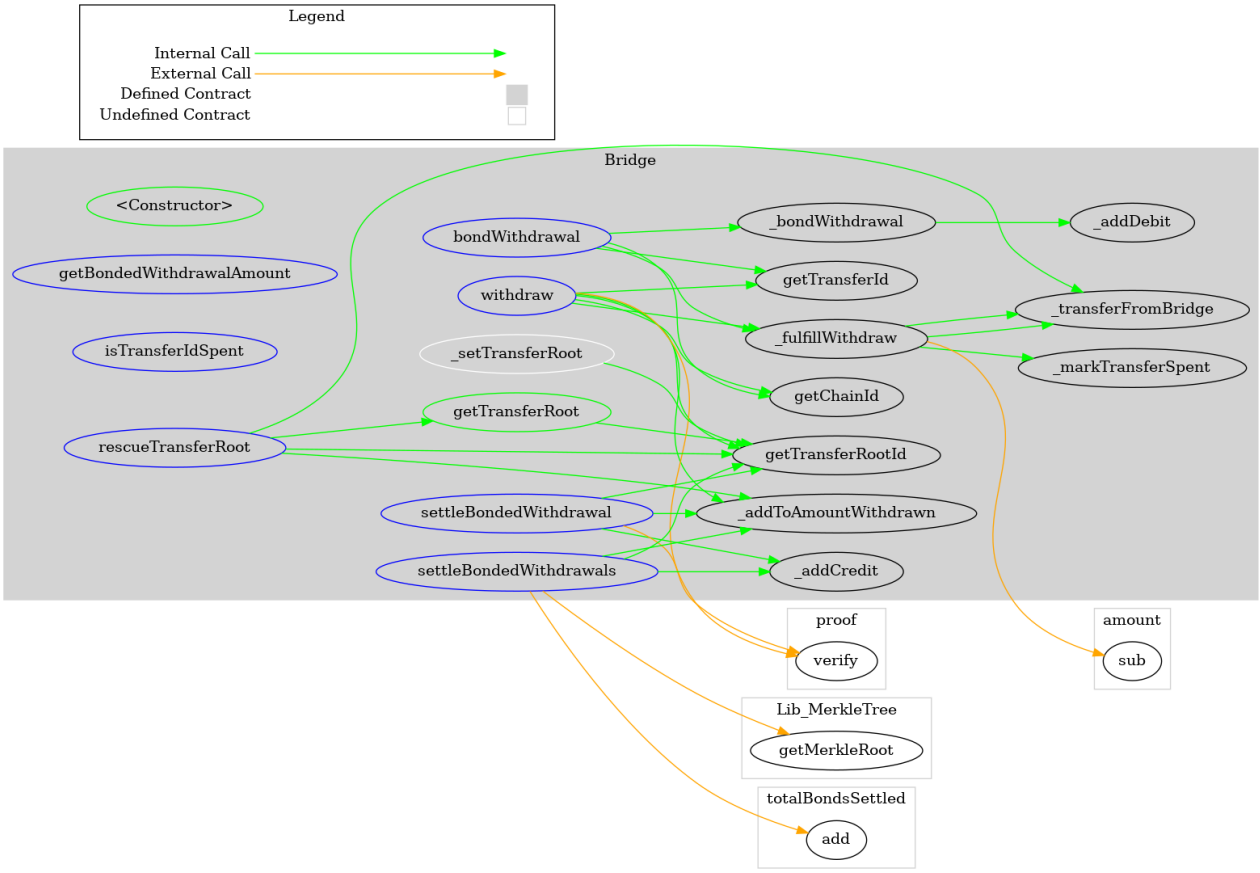


Bridges graph

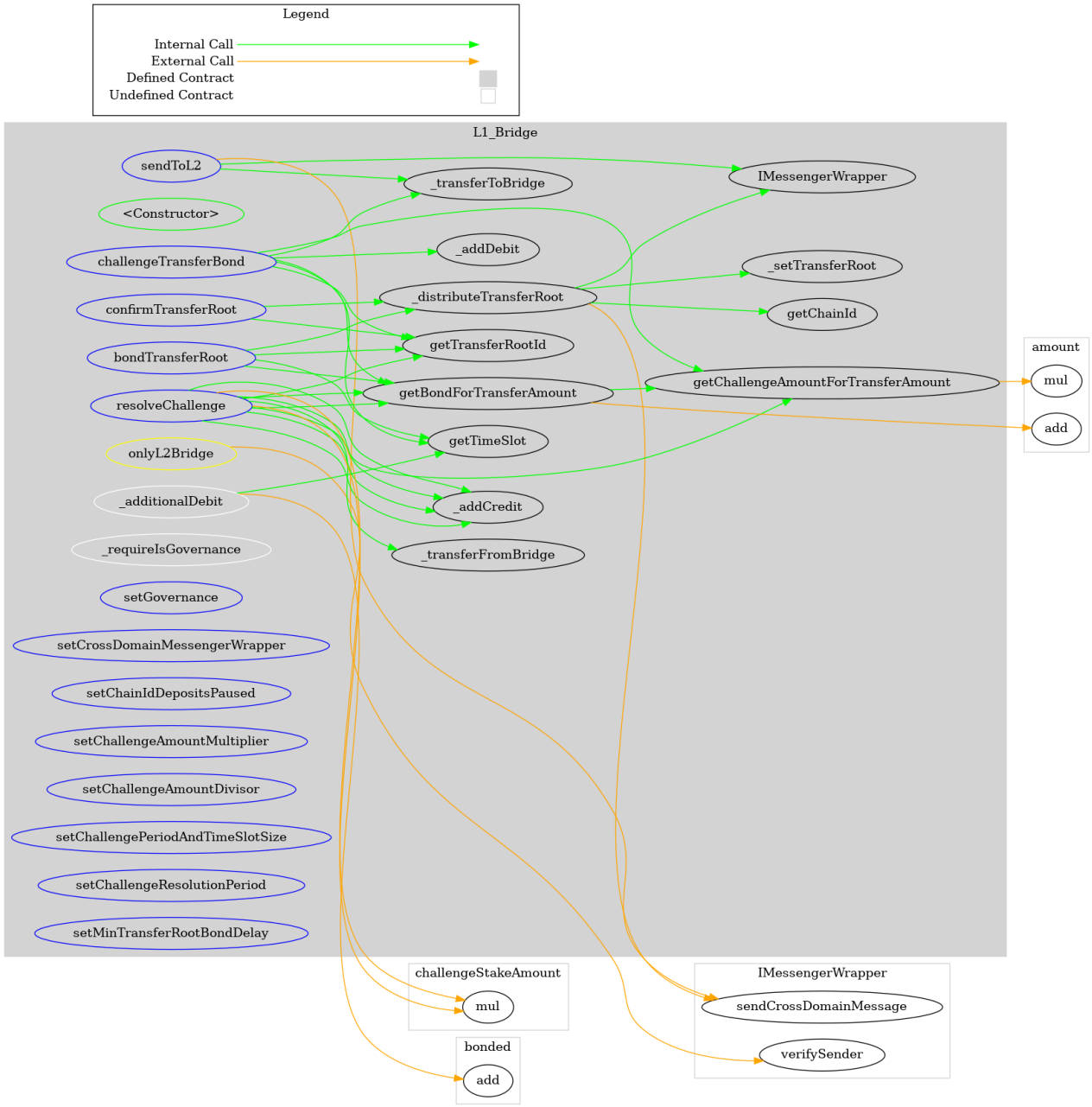




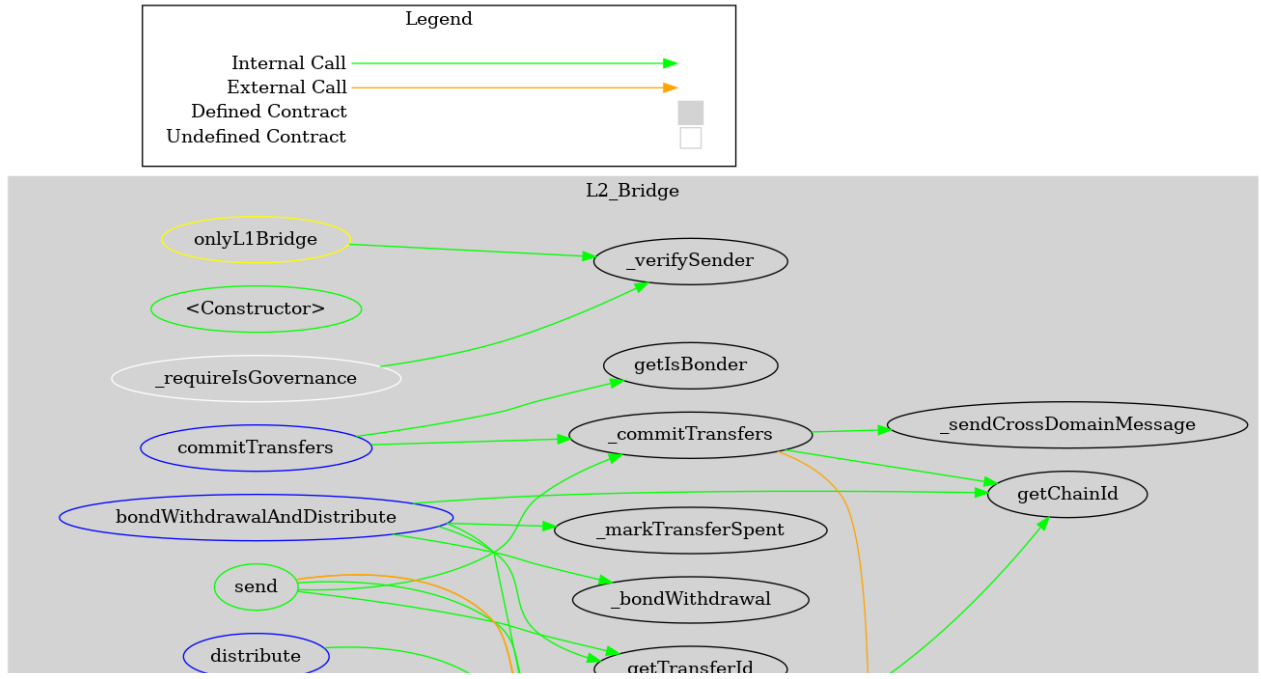
Bridge graph

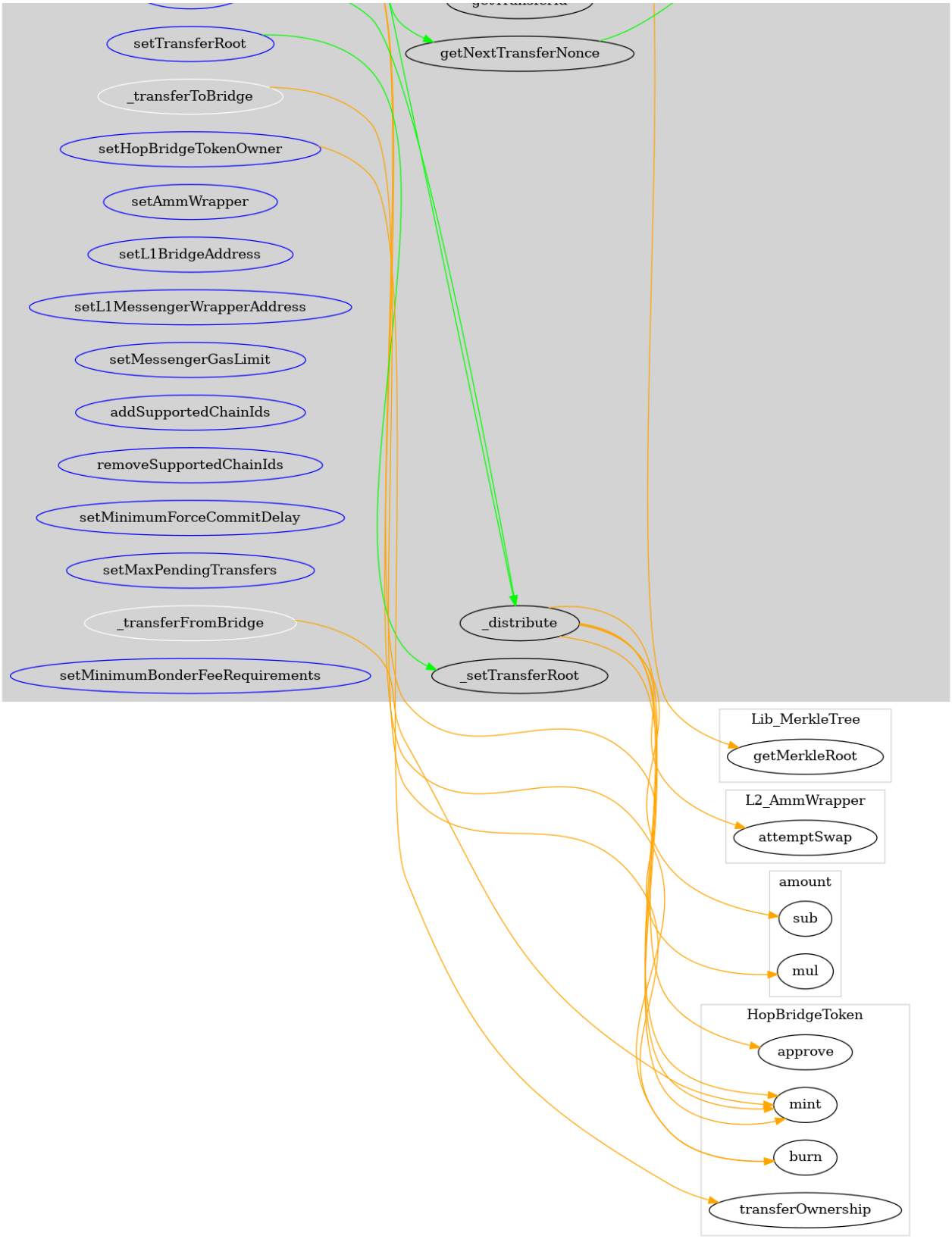

Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Bridge
<Constructor>
getBondedWithdrawalAmount
isTransferIdSpent
rescueTransferRoot
bondWithdrawal
withdraw
_setTransferRoot
getTransferRoot
settleBondedWithdrawal
settleBondedWithdrawals
_bondWithdrawal
getTransferId
_fulfillWithdraw
getChainId
getTransferRootId
_addToAmountWithdrawn
_addCredit
_addDebit
_transferFromBridge
_markTransferSpent
proof
verify
amount
sub
Lib_MerkleTree
getMerkleRoot
totalBondsSettled
add

L1_Bridge


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
L1_Bridge
sendToL2
<Constructor>
challengeTransferBond
confirmTransferRoot
bondTransferRoot
resolveChallenge
onlyL2Bridge
_additionalDebit
_requireIsGovernance
setGovernance
setCrossDomainMessengerWrapper
setChainIdDepositsPaused
setChallengeAmountMultiplier
setChallengeAmountDivisor
setChallengePeriodAndTimeSlotSize
setChallengeResolutionPeriod
setMinTransferRootBondDelay
_transferToBridge
_addDebit
_distributeTransferRoot
getTransferRootId
getBondForTransferAmount
getTimeSlot
_addCredit
_transferFromBridge
IMessengerWrapper
_setTransferRoot
getChainId
getChallengeAmountForTransferAmount
amount
mul
add
challengeStakeAmount
mul
bonded
add
IMessengerWrapper
sendCrossDomainMessage
verifySender

L2_Bridge


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
L2_Bridge
onlyL1Bridge
<Constructor>
_requireIsGovernance
commitTransfers
bondWithdrawalAndDistribute
send
distribute
_verifySender
getIsBonder
_commitTransfers
_markTransferSpent
_bondWithdrawal
getTransferId
_sendCrossDomainMessage
getChainId



Describe

Files Description Table

File Name	SHA-1 Hash
./contracts/bridges/Accounting.sol	dd830703b1d1fa5ed53b50d807ef19657c
./contracts/bridges/Bridge.sol	7a5a673d36e65640d01a3b9057cd878ec
./contracts/bridges/HopBridgeToken.sol	a006bd5671b05bce010e0583ff7371a55b
./contracts/bridges/L1_Bridge.sol	a96db5f564635ad532909560480bc079a
./contracts/bridges/L1_ERC20_Bridge.sol	d24c55c6668d4d280290f2580a8c16cb3b
./contracts/bridges/L1_ETH_Bridge.sol	242b48dad540ad06320e545905d7aa7c7
./contracts/bridges/L2_AmmWrapper.sol	53286765e1cd505c2055cc357862f9225a
./contracts/bridges/L2_ArbitrumBridge.sol	146eb9cc5903c55133e63d72b80200be4
./contracts/bridges/L2_Bridge.sol	3840324c973bd5a0b338f9e85b8f115f64a
./contracts/bridges/L2_OptimismBridge.sol	945ae3984a17b087889de7245688825bc
./contracts/bridges/L2_UniswapWrapper.sol	67a4f013266ff156636184124bc20b8db3a
./contracts/bridges/L2_XDaiBridge.sol	56d4e2b02bbb7b4d76dd31c1a47642fc5a

Contracts Description Table

Contract	Type	Bases	
└	**Function Name**	**Visibility**	**Muta**
Accounting	Implementation		
└		Public ❗	🛑
└	_transferFromBridge	Internal 🔒	🛑
└	_transferToBridge	Internal 🔒	🛑
└	_requireIsGovernance	Internal 🔒	🛑
└	_additionalDebit	Internal 🔒	
└	getIsBonder	Public ❗	
└	getCredit	Public ❗	
└	getRawDebit	External ❗	
└	getDebitAndAdditionalDebit	Public ❗	
└	stake	External ❗	💵
└	unstake	External ❗	🛑
└	addBonder	External ❗	🛑
└	removeBonder	External ❗	🛑
└	_addCredit	Internal 🔒	🛑
└	_addDebit	Internal 🔒	🛑
Bridge	Implementation	Accounting	
└		Public ❗	🛑
└	getTransferId	Public ❗	
└	getChainId	Public ❗	
└	getTransferRootId	Public ❗	
└	getTransferRoot	Public ❗	
└	getBondedWithdrawalAmount	External ❗	

Contract	Type	Bases	
└	isTransferIdSpent	External ❗	
└	withdraw	External ❗	🛑
└	bondWithdrawal	External ❗	🛑
└	settleBondedWithdrawal	External ❗	🛑
└	settleBondedWithdrawals	External ❗	🛑
└	rescueTransferRoot	External ❗	🛑
└	_markTransferSpent	Internal 🔒	🛑
└	_addToAmountWithdrawn	Internal 🔒	🛑
└	_setTransferRoot	Internal 🔒	🛑
└	_bondWithdrawal	Internal 🔒	🛑
└	_fulfillWithdraw	Private 🔐	🛑
HopBridgeToken	Implementation	ERC20, Ownable	
└		Public ❗	🛑
└	mint	External ❗	🛑
└	burn	External ❗	🛑
L1_Bridge	Implementation	Bridge	
└		Public ❗	🛑
└	sendToL2	External ❗	💵
└	bondTransferRoot	External ❗	🛑

Contract	Type	Bases	
└	confirmTransferRoot	External ❗	🛑
└	_distributeTransferRoot	Internal 🔒	🛑
└	challengeTransferBond	External ❗	💵
└	resolveChallenge	External ❗	🛑
└	_additionalDebit	Internal 🔒	
└	_requireIsGovernance	Internal 🔒	🛑
└	setGovernance	External ❗	🛑
└	setCrossDomainMessengerWrapper	External ❗	🛑
└	setChainIdDepositsPaused	External ❗	🛑
└	setChallengeAmountMultiplier	External ❗	🛑
└	setChallengeAmountDivisor	External ❗	🛑
└	setChallengePeriodAndTimeSlotSize	External ❗	🛑
└	setChallengeResolutionPeriod	External ❗	🛑
└	setMinTransferRootBondDelay	External ❗	🛑
└	getBondForTransferAmount	Public ❗	
└	getChallengeAmountForTransferAmount	Public ❗	
└	getTimeSlot	Public ❗	
L1_ERC20_Bridge	Implementation	L1_Bridge	
└		Public ❗	🛑
└	_transferFromBridge	Internal 🔒	🛑

Contract	Type	Bases	
└	_transferToBridge	Internal 🔒	🛑
L1_ETH_Bridge	Implementation	L1_Bridge	
└		Public ❗	🛑
└	_transferFromBridge	Internal 🔒	🛑
└	_transferToBridge	Internal 🔒	🛑
L2_AmmWrapper	Implementation		
└		Public ❗	🛑
└	swapAndSend	Public ❗	💵
└	attemptSwap	External ❗	🛑
L2_ArbitrumBridge	Implementation	L2_Bridge	
└		Public ❗	🛑
└	_sendCrossDomainMessage	Internal 🔒	🛑
└	_verifySender	Internal 🔒	🛑
└	setMessenger	External ❗	🛑
L2_Bridge	Implementation	Bridge	
└		Public ❗	🛑
└	_sendCrossDomainMessage	Internal 🔒	🛑
└	_verifySender	Internal 🔒	🛑
└	send	Public ❗	🛑
└	commitTransfers	External ❗	🛑
└	distribute	External ❗	🛑
└	bondWithdrawalAndDistribute	External ❗	🛑
└	setTransferRoot	External ❗	🛑

Contract	Type	Bases	
└	_commitTransfers	Internal 🔒	🛑
└	_distribute	Internal 🔒	🛑
└	_transferFromBridge	Internal 🔒	🛑
└	_transferToBridge	Internal 🔒	🛑
└	_requireIsGovernance	Internal 🔒	🛑
└	setAmmWrapper	External ❗	🛑
└	setL1BridgeAddress	External ❗	🛑
└	setL1MessengerWrapperAddress	External ❗	🛑
└	setMessengerGasLimit	External ❗	🛑
└	addSupportedChainIds	External ❗	🛑
└	removeSupportedChainIds	External ❗	🛑
└	setMinimumForceCommitDelay	External ❗	🛑
└	setMaxPendingTransfers	External ❗	🛑
└	setHopBridgeTokenOwner	External ❗	🛑
└	setMinimumBonderFeeRequirements	External ❗	🛑
└	getNextTransferNonce	Public ❗	
L2_OptimismBridge	Implementation	L2_Bridge	
└		Public ❗	🛑
└	_sendCrossDomainMessage	Internal 🔒	🛑
└	_verifySender	Internal 🔒	🛑

Contract	Type	Bases	
└	setMessenger	External ❗	🛑
└	setDefaultGasLimit	External ❗	🛑
L2_UniswapWrapper	Implementation		
└		Public ❗	🛑
└	swapAndSend	Public ❗	💵
└	attemptSwap	External ❗	🛑
└	_getHCPath	Private 🔐	
└	_getCHPath	Private 🔐	
L2_XDaiBridge	Implementation	L2_Bridge	
└		Public ❗	🛑
└	_sendCrossDomainMessage	Internal 🔒	🛑
└	_verifySender	Internal 🔒	🛑
└	setMessenger	External ❗	🛑

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Accounting

```
$ npx surya describe ./contracts/bridges/Accounting.sol

 +  Accounting
    - [Pub] <Constructor> #
    - [Int] _transferFromBridge #
    - [Int] _transferToBridge #
    - [Int] _requireIsGovernance #
    - [Int] _additionalDebit
    - [Pub] getIsBonder
    - [Pub] getCredit
```

```
    - [Ext] getRawDebit
    - [Pub] getDebitAndAdditionalDebit
    - [Ext] stake ($)
    - [Ext] unstake #
       - modifiers: requirePositiveBalance
    - [Ext] addBonder #
       - modifiers: onlyGovernance
    - [Ext] removeBonder #
       - modifiers: onlyGovernance
    - [Int] _addCredit #
    - [Int] _addDebit #


 ($) = payable function
 # = non-constant function
```

Bridge

```
$ npx surya describe ./contracts/bridges/Bridge.sol
 +  Bridge (Accounting)
    - [Pub] <Constructor> #
       - modifiers: Accounting
    - [Pub] getTransferId
    - [Pub] getChainId
    - [Pub] getTransferRootId
    - [Pub] getTransferRoot
    - [Ext] getBondedWithdrawalAmount
    - [Ext] isTransferIdSpent
    - [Ext] withdraw #
    - [Ext] bondWithdrawal #
       - modifiers: onlyBonder,requirePositiveBalance
    - [Ext] settleBondedWithdrawal #
    - [Ext] settleBondedWithdrawals #
    - [Ext] rescueTransferRoot #
       - modifiers: onlyGovernance
    - [Int] _markTransferSpent #
    - [Int] _addToAmountWithdrawn #
    - [Int] _setTransferRoot #
    - [Int] _bondWithdrawal #
    - [Prv] _fulfillWithdraw #


 ($) = payable function
 # = non-constant function
```

L2_Bridge

```
$ npx surya describe ./contracts/bridges/L2_Bridge.sol
npx: installed 63 in 1.792s
 +  L2_Bridge (Bridge)
```

```
    - [Pub] <Constructor> #
       - modifiers: Bridge
    - [Int] _sendCrossDomainMessage #
    - [Int] _verifySender #
    - [Pub] send #
    - [Ext] commitTransfers #
    - [Ext] distribute #
       - modifiers: onlyL1Bridge
    - [Ext] bondWithdrawalAndDistribute #
       - modifiers: onlyBonder,requirePositiveBalance
    - [Ext] setTransferRoot #
       - modifiers: onlyL1Bridge
    - [Int] _commitTransfers #
    - [Int] _distribute #
    - [Int] _transferFromBridge #
    - [Int] _transferToBridge #
    - [Int] _requireIsGovernance #
    - [Ext] setAmmWrapper #
       - modifiers: onlyGovernance
    - [Ext] setL1BridgeAddress #
       - modifiers: onlyGovernance
    - [Ext] setL1MessengerWrapperAddress #
       - modifiers: onlyGovernance
    - [Ext] setMessengerGasLimit #
       - modifiers: onlyGovernance
    - [Ext] addSupportedChainIds #
       - modifiers: onlyGovernance
    - [Ext] removeSupportedChainIds #
       - modifiers: onlyGovernance
    - [Ext] setMinimumForceCommitDelay #
       - modifiers: onlyGovernance
    - [Ext] setMaxPendingTransfers #
       - modifiers: onlyGovernance
    - [Ext] setHopBridgeTokenOwner #
       - modifiers: onlyGovernance
    - [Ext] setMinimumBonderFeeRequirements #
       - modifiers: onlyGovernance
    - [Pub] getNextTransferNonce


 ($) = payable function
 # = non-constant function
```

L1_Bridge

```
$ npx surya describe ./contracts/bridges/L1_Bridge.sol
npx: installed 63 in 3.544s
 +  L1_Bridge (Bridge)
    - [Pub] <Constructor> #
       - modifiers: Bridge
    - [Ext] sendToL2 ($)
    - [Ext] bondTransferRoot #
```

```
       - modifiers: onlyBonder,requirePositiveBalance
    - [Ext] confirmTransferRoot #
       - modifiers: onlyL2Bridge
    - [Int] _distributeTransferRoot #
    - [Ext] challengeTransferBond ($)
    - [Ext] resolveChallenge #
    - [Int] _additionalDebit
    - [Int] _requireIsGovernance #
    - [Ext] setGovernance #
       - modifiers: onlyGovernance
    - [Ext] setCrossDomainMessengerWrapper #
       - modifiers: onlyGovernance
    - [Ext] setChainIdDepositsPaused #
       - modifiers: onlyGovernance
    - [Ext] setChallengeAmountMultiplier #
       - modifiers: onlyGovernance
    - [Ext] setChallengeAmountDivisor #
       - modifiers: onlyGovernance
    - [Ext] setChallengePeriodAndTimeSlotSize #
       - modifiers: onlyGovernance
    - [Ext] setChallengeResolutionPeriod #
       - modifiers: onlyGovernance
    - [Ext] setMinTransferRootBondDelay #
       - modifiers: onlyGovernance
    - [Pub] getBondForTransferAmount
    - [Pub] getChallengeAmountForTransferAmount
    - [Pub] getTimeSlot


 ($) = payable function
 # = non-constant function
```

L2_AmmWrapper

```
$ npx surya describe ./contracts/bridges/L2_AmmWrapper.sol
npx: installed 63 in 2.339s
 +  L2_AmmWrapper
    - [Pub] <Constructor> #
    - [Pub] swapAndSend ($)
    - [Ext] attemptSwap #


 ($) = payable function
 # = non-constant function
```

ArbitrumMessengerWrapper

```
$ npx surya describe ./contracts/wrappers/ArbitrumMessengerWrapper.sol
npx: installed 63 in 2.389s
 +  ArbitrumMessengerWrapper (MessengerWrapper)
    - [Pub] <Constructor> #
```

```
    - [Pub] sendCrossDomainMessage #
       - modifiers: onlyL1Bridge
    - [Pub] verifySender #


 ($) = payable function
 # = non-constant function
```

OptimismMessengerWrapper

```
$ npx surya describe ./contracts/./wrappers/OptimismMessengerWrapper.sol
npx: installed 63 in 3.061s
 +  OptimismMessengerWrapper (MessengerWrapper)
    - [Pub] <Constructor> #
    - [Pub] sendCrossDomainMessage #
       - modifiers: onlyL1Bridge
    - [Pub] verifySender #


 ($) = payable function
 # = non-constant function
```

XDaiMessengerWrapper

```
$ npx surya describe ./contracts/./wrappers/XDaiMessengerWrapper.sol
npx: installed 63 in 4.416s
 +  XDaiMessengerWrapper (MessengerWrapper)
    - [Pub] <Constructor> #
    - [Pub] sendCrossDomainMessage #
       - modifiers: onlyL1Bridge
    - [Pub] verifySender #


 ($) = payable function
 # = non-constant function
```

License