LAYER TWO AMSTERDAM

DEVCONNECT AMS

L2BEAT

Hadrien Croubois

Going Cross-Chain

with OpenZeppelin

STARKWARE / ARBITRUM / OPTIMISM / BOBA NETWORK / METIS / FUEL LABS / NERVOS / ZKSYNC

OpenZeppelin

Our mission is to protect the open economy.

OpenZeppelin is a Web3 cybersecurity company that provides **security audits** and **products** for decentralized systems.

Projects of all sizes — from new startups to established organizations — trust OpenZeppelin to build, inspect, and connect to Web3.

Join the team!

Compound coinbase AAVE

δY/δX brave UMA

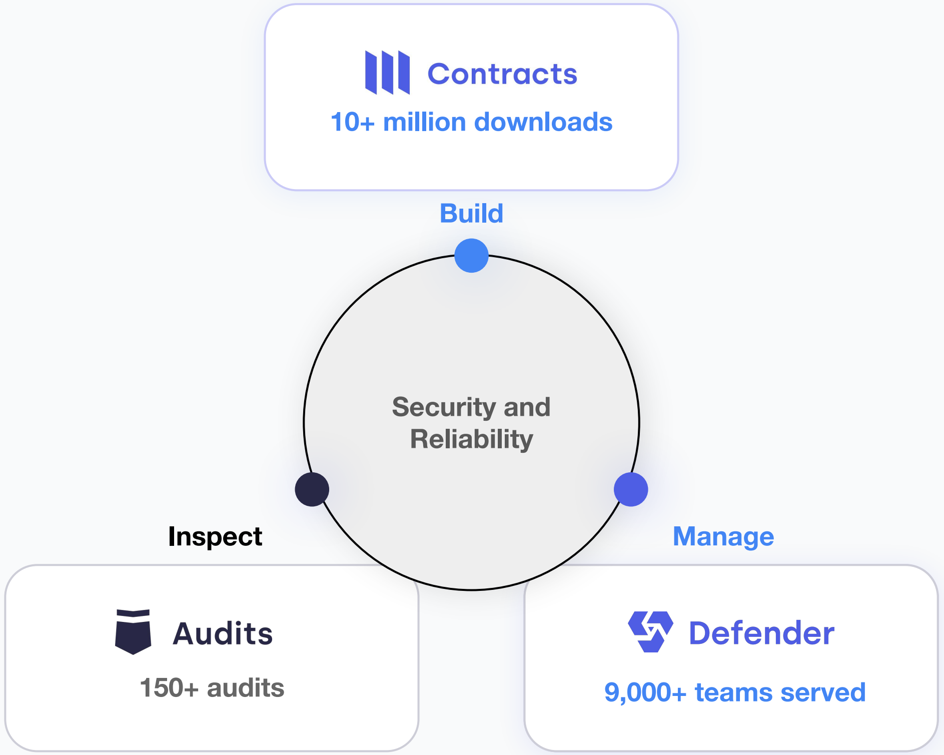
Balancer ethereum foundation augur

celo CIRCLE BitGo

pool together Set opyn

Security, Reliability and Risk Management

OpenZeppelin provides a complete suite of **security and reliability products** to build, manage, and inspect all aspects of software development and operations for Ethereum projects.



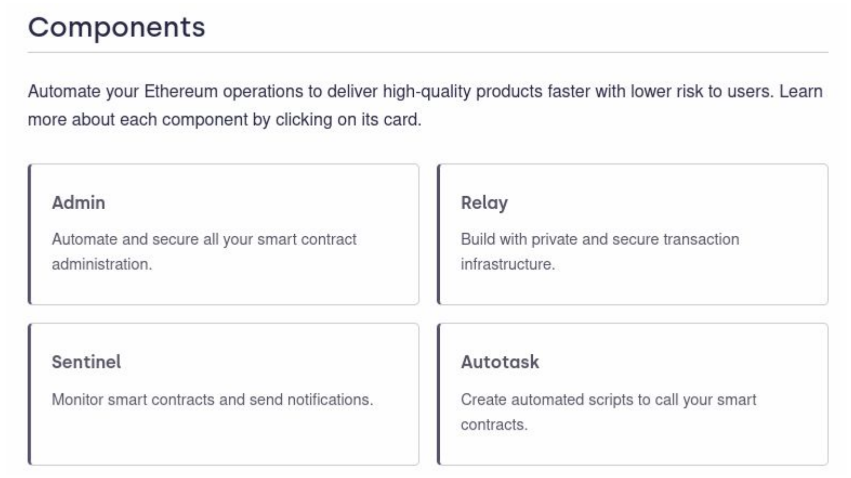
Defender

A platform to automate Ethereum operations and deliver high-quality products faster.

- ✓ Automate your smart contract administration with a clean UI.
- ✓ Build with private and secure transaction infrastructure.
- ✓ Create automated scripts to call your smart contracts.
- ✓ Quickly implement security best practices.

VISIT SITE GO TO DOCS

Cross-Chain Operations with OpenZeppelin Defender



Components

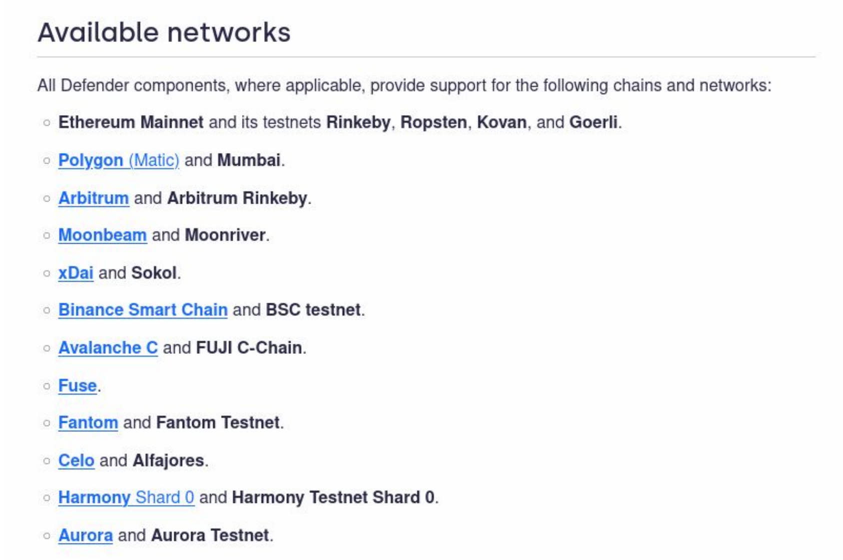
Automate your Ethereum operations to deliver high-quality products faster with lower risk to users. Learn more about each component by clicking on its card.

Admin
Automate and secure all your smart contract administration.

Relay
Build with private and secure transaction infrastructure.

Sentinel
Monitor smart contracts and send notifications.

Autotask
Create automated scripts to call your smart contracts.



Available networks

All Defender components, where applicable, provide support for the following chains and networks:

- **Ethereum Mainnet** and its testnets **Rinkeby**, **Ropsten**, **Kovan**, and **Goerli**.
- **Polygon** (Matic) and **Mumbai**.
- **Arbitrum** and **Arbitrum Rinkeby**.
- **Moonbeam** and **Moonriver**.
- **xDai** and **Sokol**.
- **Binance Smart Chain** and **BSC testnet**.
- **Avalanche C** and **FUJI C-Chain**.
- **Fuse**.
- **Fantom** and **Fantom Testnet**.
- **Celo** and **Alfajores**.
- **Harmony** Shard 0 and **Harmony Testnet Shard 0**.
- **Aurora** and **Aurora Testnet**.

Contracts

A library of modular, reusable, secure smart contracts for the Ethereum network, written in Solidity.

- ✓ Leverage standard, tested, and community-reviewed contracts.
- ✓ Most popular library in the industry.
- ✓ Learn from best practices adopted by the ecosystem.
- ✓ Reduce your attack surface by reusing audited code.

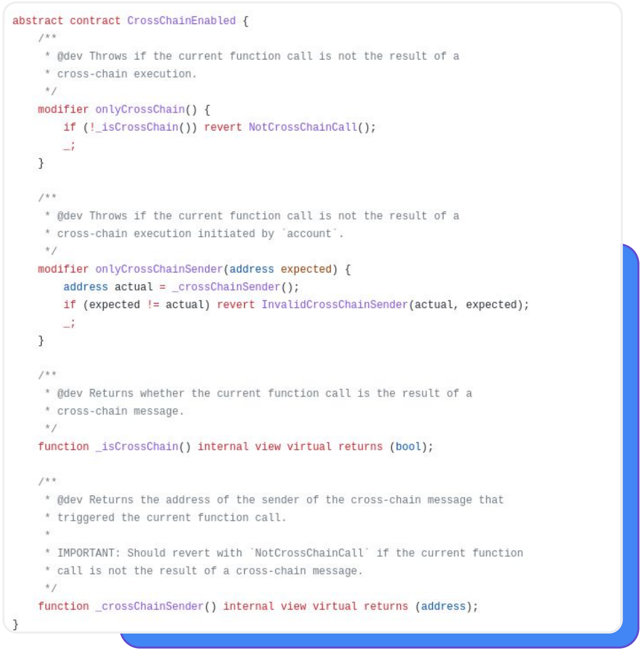
VISIT SITE GO TO DOCS

New in 4.6: Cross-Chain Enabled

Goal: An abstraction to build cross-chain aware contracts. Includes internal functions and modifiers to restrict cross chain operations.

Implementations available for:

- AMB (gnosis chain)
- Arbitrum
- Optimism
- Polygon (Root → Child)



```solidity
abstract contract CrossChainEnabled {
    /**
     * @dev Throws if the current function call is not the result of a
     * cross-chain execution.
     */
    modifier onlyCrossChain() {
        if (!_isCrossChain()) revert NotCrossChainCall();
        _;
    }

    /**
     * @dev Throws if the current function call is not the result of a
     * cross-chain execution initiated by `account`.
     */
    modifier onlyCrossChainSender(address expected) {
        address actual = _crossChainSender();
        if (expected != actual) revert InvalidCrossChainSender(actual, expected);
        _;
    }

    /**
     * @dev Returns whether the current function call is the result of a
     * cross-chain message.
     */
    function _isCrossChain() internal view virtual returns (bool);

    /**
     * @dev Returns the address of the sender of the cross-chain message that
     * triggered the current function call.
     *
     * IMPORTANT: Should revert with `NotCrossChainCall` if the current function
     * call is not the result of a cross-chain message.
     */
    function _crossChainSender() internal view virtual returns (address);
}
```

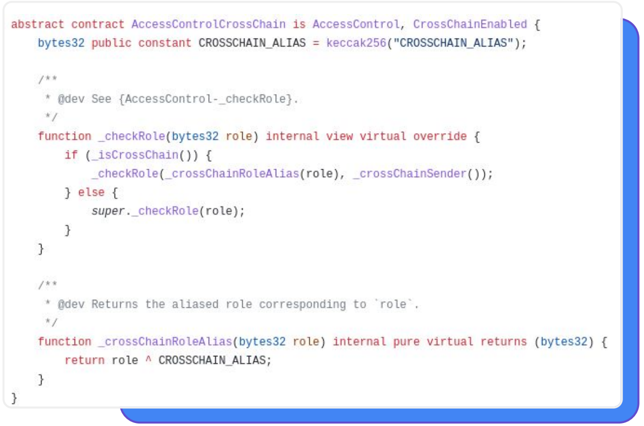
New in 4.6: Cross-Chain: AccessControl

Goal: Access Control extension to support restriction to cross-chain addresses.

For each role, such as the `MINTER_ROLE`, there are two variants:

- The "normal" version
- The "aliased" version

In order to call an `onlyRole(MINTER_ROLE)` function from a cross-chain call, the caller must have the aliased role.



```
abstract contract AccessControlCrossChain is AccessControl, CrossChainEnabled {
    bytes32 public constant CROSSCHAIN_ALIAS = keccak256("CROSSCHAIN_ALIAS");

    /**
     * @dev See {AccessControl-_checkRole}.
     */
    function _checkRole(bytes32 role) internal view virtual override {
        if (_isCrossChain()) {
            _checkRole(_crossChainRoleAlias(role), _crossChainSender());
        } else {
            super._checkRole(role);
        }
    }

    /**
     * @dev Returns the aliased role corresponding to `role`.
     */
    function _crossChainRoleAlias(bytes32 role) internal pure virtual returns (bytes32) {
        return role ^ CROSSCHAIN_ALIAS;
    }
}
```

Abstract contract vs Library

Cross-chain support is available through:

- Inheritance / Contracts (specialization of the CrossChainEnabled framework)
- Composition / Libraries (with a standard interface)

Libraries are more difficult to use but enable more powerful patterns.

Idea: A contract that can receive messages from multiple bridges, identify the bridge, and recover the sender transparently.

Issue: Some chains (polygon) need specific public functions at the contract level to receive cross-chain calls. This cannot be abstracted in a library.

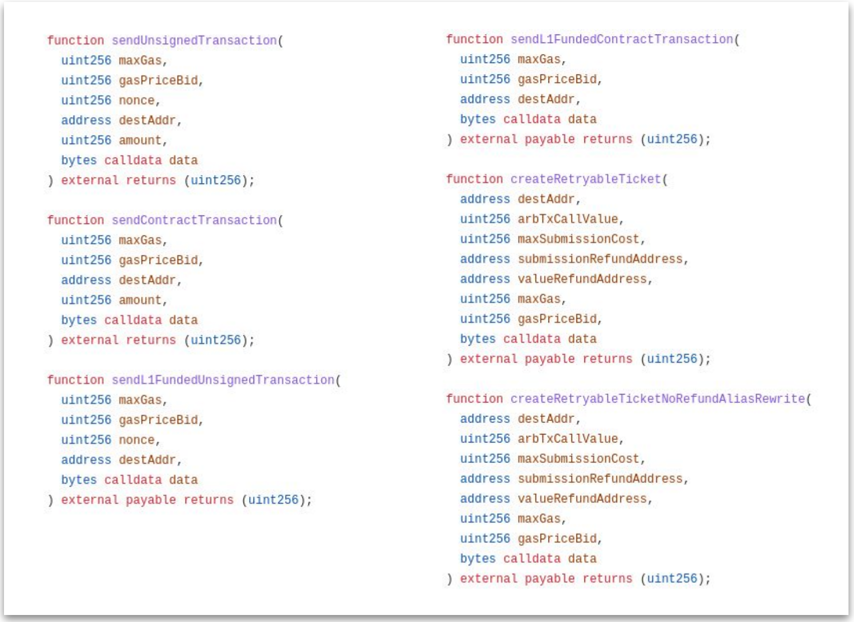
On the Roadmap: Emitting Cross-Chain Calls

Available:	Receiving call and identifying the sender
Next:	Emitting calls to other chains

Emitting is way more difficult to standardize than receiving.

- Some chains require many parameters that are difficult to abstract.
- There often is not any "default" value observable on-chain. Forwarding them is painful, particularly with delayed execution (governor/timelock).
- **"Less is more"** should be your guideline when designing these systems.

On the Roadmap: Emitting Cross-Chain Calls



```
function sendUnsignedTransaction(
  uint256 maxGas,
  uint256 gasPriceBid,
  uint256 nonce,
  address destAddr,
  uint256 amount,
  bytes calldata data
) external returns (uint256);

function sendContractTransaction(
  uint256 maxGas,
  uint256 gasPriceBid,
  address destAddr,
  uint256 amount,
  bytes calldata data
) external returns (uint256);

function sendL1FundedUnsignedTransaction(
  uint256 maxGas,
  uint256 gasPriceBid,
  uint256 nonce,
  address destAddr,
  bytes calldata data
) external payable returns (uint256);
```

```
function sendL1FundedContractTransaction(
  uint256 maxGas,
  uint256 gasPriceBid,
  address destAddr,
  bytes calldata data
) external payable returns (uint256);

function createRetryableTicket(
  address destAddr,
  uint256 arbTxCallValue,
  uint256 maxSubmissionCost,
  address submissionRefundAddress,
  address valueRefundAddress,
  uint256 maxGas,
  uint256 gasPriceBid,
  bytes calldata data
) external payable returns (uint256);

function createRetryableTicketNoRefundAliasRewrite(
  address destAddr,
  uint256 arbTxCallValue,
  uint256 maxSubmissionCost,
  address submissionRefundAddress,
  address valueRefundAddress,
  uint256 maxGas,
  uint256 gasPriceBid,
  bytes calldata data
) external payable returns (uint256);
```

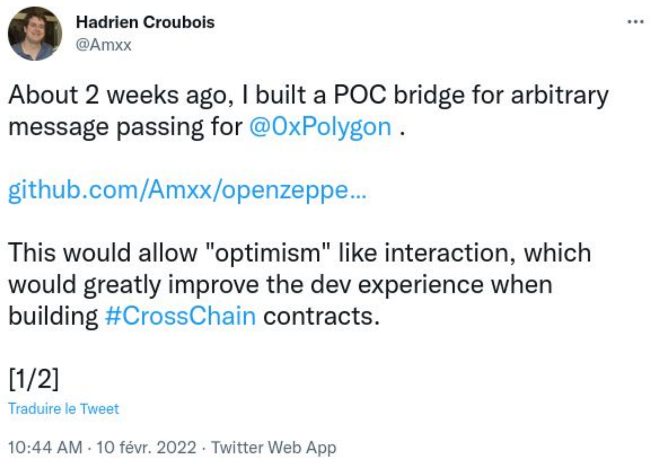
We need more standards: Bridges

Some things can be fixed in the user space:

- Sender recovery
- Retryable tickets

Personal opinion:

- We need standard interfaces for arbitrary message passing. “AMB” or “CrossChainMessenger” style.
- Easy to build on top of existing mechanisms.
- Reduces the complexity of user contracts.
- Hard to standardize, should be a community effort!



Hadrien Croubois
@Amxx

About 2 weeks ago, I built a POC bridge for arbitrary message passing for @0xPolygon .

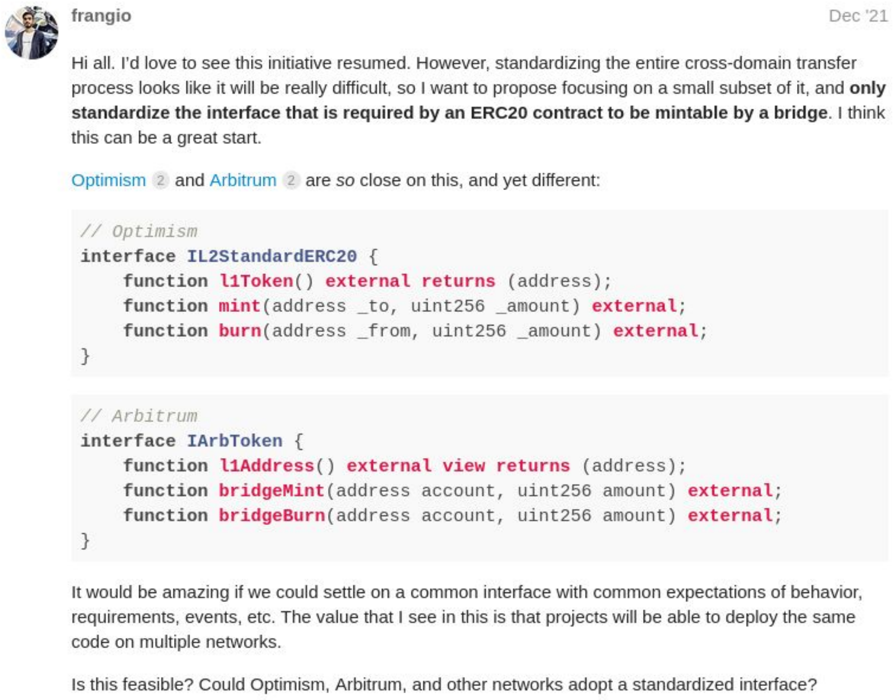
github.com/Amxx/openzeppe...

This would allow "optimism" like interaction, which would greatly improve the dev experience when building #CrossChain contracts.

[1/2]
Traduire le Tweet

10:44 AM · 10 févr. 2022 · Twitter Web App

We need more standards: ERC20 extension



frangio Dec '21

Hi all. I'd love to see this initiative resumed. However, standardizing the entire cross-domain transfer process looks like it will be really difficult, so I want to propose focusing on a small subset of it, and **only standardize the interface that is required by an ERC20 contract to be mintable by a bridge**. I think this can be a great start.

Optimism 2 and Arbitrum 2 are *so* close on this, and yet different:

```
// Optimism
interface IL2StandardERC20 {
    function l1Token() external returns (address);
    function mint(address _to, uint256 _amount) external;
    function burn(address _from, uint256 _amount) external;
}
```

```
// Arbitrum
interface IArbToken {
    function l1Address() external view returns (address);
    function bridgeMint(address account, uint256 amount) external;
    function bridgeBurn(address account, uint256 amount) external;
}
```

It would be amazing if we could settle on a common interface with common expectations of behavior, requirements, events, etc. The value that I see in this is that projects will be able to deploy the same code on multiple networks.

Is this feasible? Could Optimism, Arbitrum, and other networks adopt a standardized interface?

Beyond Solidity: OpenZeppelin Contracts for Cairo

Announcing OpenZeppelin Contracts for Cairo

APRIL 5, 2022 | IN ANNOUNCEMENTS | BY MARTIN TRIAY

OpenZeppelin Contracts, the standard for smart contract development in Solidity, is now expanding to the Cairo language. With the new OpenZeppelin Contracts for Cairo version 0.1.0, developers can leverage standard smart contracts written in Cairo to build their own applications on top of StarkNet, a Zero Knowledge Rollup.

@openzeppelin/contracts

docs.openzeppelin.com

forum.openzeppelin.com

defender.openzeppelin.com

Thank you!

Learn more

openzeppelin.com/**contracts**

forum.openzeppelin.com

docs.openzeppelin.com

Contact

@amxx

hadrien@openzeppelin.com