



# OpenZeppelin Contracts

Version 5.0

EthCC[6] - July 2023

**Hadrien Croubois**

hadrien@openzeppelin.com

@Amxx

# OpenZeppelin's thesis

- There will be a **trillion dollar open economy** built on blockchains and **powered by smart contracts**
- This new, open economy will be **built by teams of creative people developing new applications used by billions of people**
- These teams **will need a set of tools, products and services** to make sure that what they are building is **safe and reliable**
- OpenZeppelin will be a **leading provider of these solutions**, allowing teams to **build faster with lower risk**

# OpenZeppelin's products



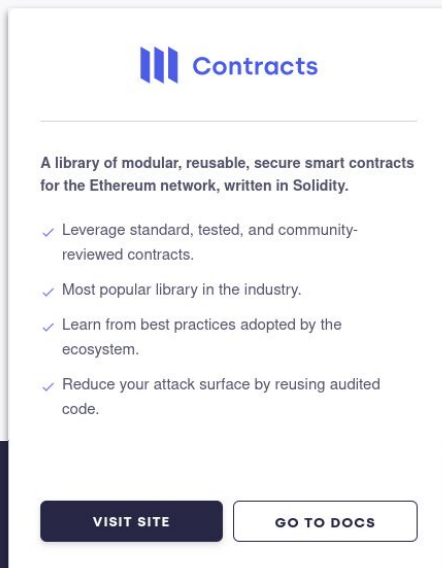
L1 / L2 Networks



# Contracts

@openzeppelin/contracts@4.9.2

@openzeppelin/contracts-upgradeable@4.9.2



The screenshot shows the top section of the OpenZeppelin Contracts GitHub repository page. At the top left is the OpenZeppelin logo (three vertical bars) followed by the word "Contracts". Below this is a horizontal line. The main text reads: "A library of modular, reusable, secure smart contracts for the Ethereum network, written in Solidity." Below this text is a list of four bullet points, each starting with a checkmark. At the bottom of the screenshot are two buttons: "VISIT SITE" and "GO TO DOCS".

**Contracts**

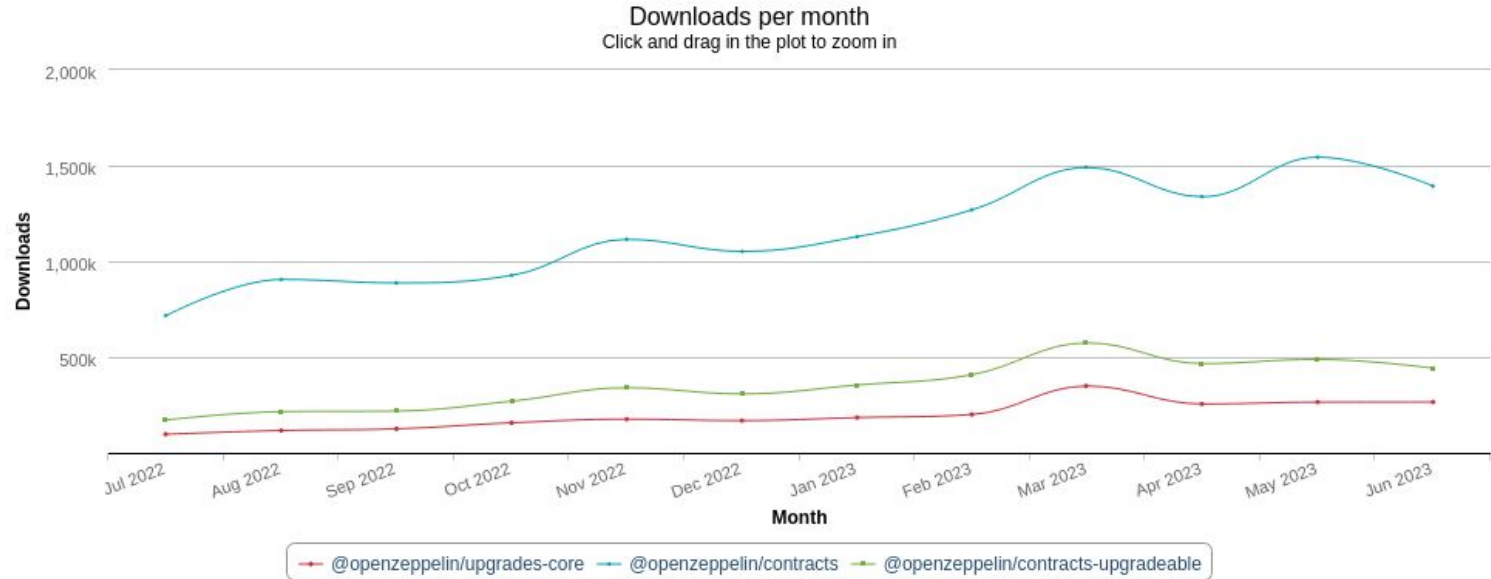
---

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](#)

# Some statistics



# Immunefi bug bounty

**OpenZeppelin** [Submit a Bug](#)

**15 November 2021**  
Live since

**No**  
KYC required

**\$25,000**  
Maximum bounty

### Program Overview

OpenZeppelin is the premier crypto cybersecurity technology and services company, trusted by the most used DeFi and NFT projects. Founded in 2015 with the mission to protect the open economy, OpenZeppelin provides a complete suite of security products to build, manage, and inspect all aspects of software development and operations for Ethereum projects.

OpenZeppelin safeguards tens of billions of dollars in funds for leading crypto organizations including Coinbase, Ethereum Foundation, Compound, Aave, TheGraph, and many others.

For more information about OpenZeppelin, please visit <https://openzeppelin.com/>.

This bug bounty program is focused on their smart contracts and is focused on preventing:

- Loss of funds by freezing or theft
- Denial of service (smart contract is made unable to operate)
- Access control is bypassed, including privilege escalation
- Smart contract does not behave as intended

This is an overlay bug bounty program for OpenZeppelin's contracts library. A vulnerability in an OpenZeppelin library would likely affect many other projects and could trigger various other bounties. This program would be potentially additive to these cases.

# One year of features

4.7.0 to 4.9.2

## Governance & Votes

- Support timestamp (see ERC-6372)

## Access

- Ownable2Step
- AccessControlDefaultAdminRules

## NFTs

- ERC721Consecutive
- ERC721Wrapper

## DeFi

- ERC4626 virtual offset

## Libraries

- ShortStrings
- eip712Domain (see EIP-5267)

## Other

- Procedural code generation
- Formal verification and fuzzing

And many many more... (see Changelog)

# @openzeppelin/merkle-tree

- **Build tree from leaf details**, with automatic double hashing
- **Dump to file / load from file**
- **Generate proofs and multiproofs**
- **Verify proofs and multiproofs**
- **Get root, render, hash leaf, ...**

## Quick Start

```
npm install @openzeppelin/merkle-tree
```

## Building a Tree

```
import { StandardMerkleTree } from "@openzeppelin/merkle-tree";
import fs from "fs";

// (1)
const values = [
  ["0x1111111111111111111111111111111111111111111111111111111111111111", "50000000000000000000000000000000"],
  ["0x2222222222222222222222222222222222222222222222222222222222222222", "25000000000000000000000000000000"]
];

// (2)
const tree = StandardMerkleTree.of(values, ["address", "uint256"]);

// (3)
console.log('Merkle Root:', tree.root);

// (4)
fs.writeFileSync("tree.json", JSON.stringify(tree.dump()));
```

1. Get the values to include in the tree. (Note: Consider reading them from a file.)
2. Build the merkle tree. Set the encoding to match the values.
3. Print the merkle root. You will probably publish this value on chain in a smart contract.
4. Write a file that describes the tree. You will distribute this to users so they can generate proofs for values in the tree.

## Obtaining a Proof

Assume we're looking to generate a proof for the entry that corresponds to address `0x11...11`.

```
import { StandardMerkleTree } from "@openzeppelin/merkle-tree";
import fs from "fs";

// (1)
const tree = StandardMerkleTree.load(JSON.parse(fs.readFileSync("tree.json")));
```



# What is coming in 5.0?

# The AccessManager system

A single contract of managing all the permissions in your dApp

**AccessManager**

and

**AccessManaged**

(abstract contract that provides a “restricted” modifier)

and

**AccessManagedAdapter**

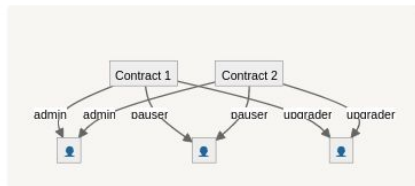
(for contracts that are Ownable or AccessControl)

Design is still work in progress.

# The AccessManager system

A single contract of managing all the permissions in your dApp

## AccessControl

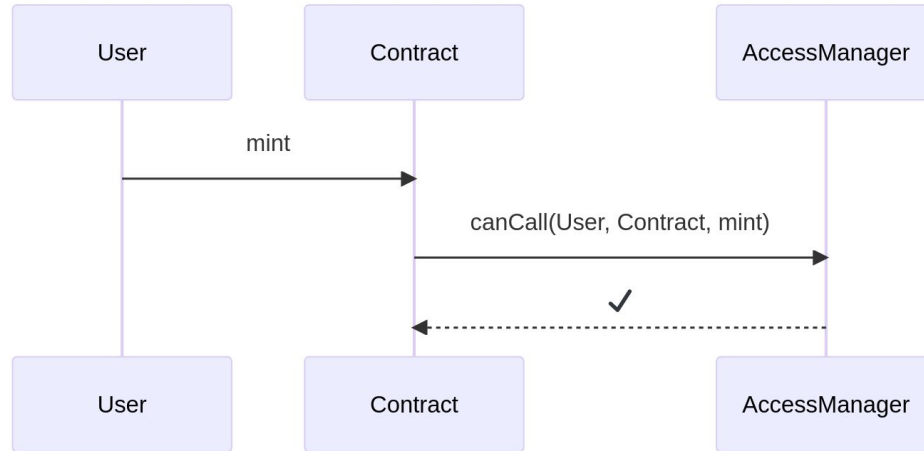


## AccessManager



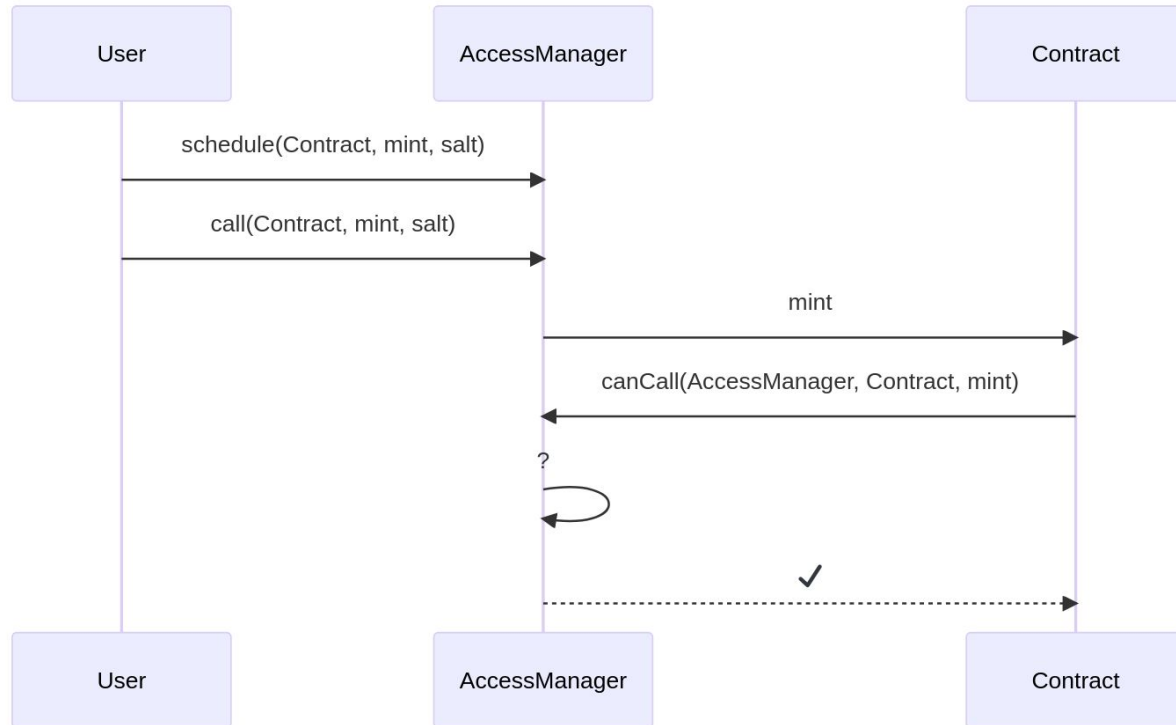
# Calling a restricted contract

Directly



# Calling a restricted contract

With a Delay



# Namespace storage

Stop worrying about inheritance ordering during upgrades

```
contract Abc {
  uint256 private x;
  address private y;

  /// @custom:oz-unsafe-allow immutable
  uint256 private immutable z;

  function _foo() internal {
    (bool success, ) = y.call{value: x}(new bytes(0));
    require(success);
  }

  function _bar() internal view returns (uint256) {
    return z;
  }
}
```

```
contract AbcUpgradeable {
  // keccak256(abi.encode(uint256(keccak256("openzeppelin.storage.Abc")) - 1)) = 0xb0276da0e97acf68f2f0987ab59d31f961df149c06dd6405568fd584af4ca1f0
  bytes32 private constant MAIN_STORAGE_LOCATION = 0xb0276da0e97acf68f2f0987ab59d31f961df149c06dd6405568fd584af4ca1f0;

  /// @custom:storage-location erc7201:openzeppelin.storage.Abc
  struct AbcStorage {
    uint256 x;
    address y;
  }

  /// @custom:oz-unsafe-allow immutable
  uint256 private immutable z;

  function _getAbcStorage() private pure returns (AbcStorage storage $) {
    bytes32 slot = MAIN_STORAGE_LOCATION;
    assembly { $.slot := slot }
  }

  function _foo() internal {
    AbcStorage storage $ = _getAbcStorage();
    (bool success, ) = $.y.call{value: $.x}(new bytes(0));
    require(success);
  }

  function _bar() internal view returns (uint256) {
    return z;
  }
}
```

Design is still work in progress.

**See ERC-7201**

Status: draft

# Contracts refactor

Breaking changes that requires a major version

## Tokens (ERC20, ERC721, ERC1155)

- Remove hooks in the token contracts
- Lock **\_transfer**, **\_mint**, **\_burn**
- Single function to override: **\_update(...)**

## Nonces

- Dedicated abstract contract

## Governor voting

- Bytes signatures
- Nonce protected
- ERC-1271 support

## Ownable

- Take initial owner as an argument



# More modern solidity

Catching up with the compiler

```
pragma solidity ^0.8.0;
import "../utils/Context.sol";

abstract contract Ownable is Context {
    address private _owner;

    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

    constructor() {
        _transferOwnership(_msgSender());
    }

    modifier onlyOwner() {
        _checkOwner();
        _;
    }

    function owner() public view virtual returns (address) {
        return _owner;
    }

    function _checkOwner() internal view virtual {
        require(owner() == _msgSender(), "Ownable: caller is not the owner");
    }

    function renounceOwnership() public virtual onlyOwner {
        _transferOwnership(address(0));
    }

    function transferOwnership(address newOwner) public virtual onlyOwner {
        require(newOwner != address(0), "Ownable: new owner is the zero address");
        _transferOwnership(newOwner);
    }

    function _transferOwnership(address newOwner) internal virtual {
        address oldOwner = _owner;
        _owner = newOwner;
        emit OwnershipTransferred(oldOwner, newOwner);
    }
}
```

```
pragma solidity ^0.8.19;
import {Context} from "../utils/Context.sol";

abstract contract Ownable is Context {
    address private _owner;

    error OwnableUnauthorizedAccount(address account);
    error OwnableInvalidOwner(address owner);
    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

    constructor(address initialOwner) {
        _transferOwnership(initialOwner);
    }

    modifier onlyOwner() {
        _checkOwner();
        _;
    }

    function owner() public view virtual returns (address) {
        return _owner;
    }

    function _checkOwner() internal view virtual {
        if (owner() != _msgSender()) {
            revert OwnableUnauthorizedAccount(_msgSender());
        }
    }

    function renounceOwnership() public virtual onlyOwner {
        _transferOwnership(address(0));
    }

    function transferOwnership(address newOwner) public virtual onlyOwner {
        if (newOwner == address(0)) {
            revert OwnableInvalidOwner(address(0));
        }
        _transferOwnership(newOwner);
    }

    function _transferOwnership(address newOwner) internal virtual {
        address oldOwner = _owner;
        _owner = newOwner;
        emit OwnershipTransferred(oldOwner, newOwner);
    }
}
```

Explicit imports, Custom errors, abi.encodeCall, bytes.concat, string.concat, ...

# More modern solidity

Catching up with the compiler

```
pragma solidity ^0.8.0;
import "../utils/Context.sol";

abstract contract Ownable is Context {
    address private _owner;

    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

    constructor() {
        _transferOwnership(_msgSender());
    }

    modifier onlyOwner() {
        _checkOwner();
        _;
    }

    function owner() public view virtual returns (address) {
        return _owner;
    }

    function _checkOwner() internal view virtual {
        require(owner() == _msgSender(), "Ownable: caller is not the owner");
    }

    function renounceOwnership() public virtual onlyOwner {
        _transferOwnership(address(0));
    }

    function transferOwnership(address newOwner) public virtual onlyOwner {
        require(newOwner != address(0), "Ownable: new owner is the zero address");
        _transferOwnership(newOwner);
    }

    function _transferOwnership(address newOwner) internal virtual {
        address oldOwner = _owner;
        _owner = newOwner;
        emit OwnershipTransferred(oldOwner, newOwner);
    }
}
```



```
pragma solidity ^0.8.19;
import {Context} from "../utils/Context.sol";

abstract contract Ownable is Context {
    address private _owner;

    error OwnableUnauthorizedAccount(address account);
    error OwnableInvalidOwner(address owner);
    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

    constructor(address initialOwner) {
        _transferOwnership(initialOwner);
    }

    modifier onlyOwner() {
        _checkOwner();
        _;
    }

    function owner() public view virtual returns (address) {
        return _owner;
    }

    function _checkOwner() internal view virtual {
        if (owner() != _msgSender()) {
            revert OwnableUnauthorizedAccount(_msgSender());
        }
    }

    function renounceOwnership() public virtual onlyOwner {
        _transferOwnership(address(0));
    }

    function transferOwnership(address newOwner) public virtual onlyOwner {
        if (newOwner == address(0)) {
            revert OwnableInvalidOwner(address(0));
        }
        _transferOwnership(newOwner);
    }

    function _transferOwnership(address newOwner) internal virtual {
        address oldOwner = _owner;
        _owner = newOwner;
        emit OwnershipTransferred(oldOwner, newOwner);
    }
}
```

Explicit imports, Custom errors, abi.encodeCall, bytes.concat, string.concat, ...

# More modern solidity

Catching up with the compiler

```
pragma solidity ^0.8.0;
import "../utils/Context.sol";

abstract contract Ownable is Context {
    address private _owner;

    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

    constructor() {
        _transferOwnership(_msgSender());
    }

    modifier onlyOwner() {
        _checkOwner();
        _;
    }

    function owner() public view virtual returns (address) {
        return _owner;
    }

    function _checkOwner() internal view virtual {
        require(owner() == _msgSender(), "Ownable: caller is not the owner");
    }

    function renounceOwnership() public virtual onlyOwner {
        _transferOwnership(address(0));
    }

    function transferOwnership(address newOwner) public virtual onlyOwner {
        require(newOwner != address(0), "Ownable: new owner is the zero address");
        _transferOwnership(newOwner);
    }

    function _transferOwnership(address newOwner) internal virtual {
        address oldOwner = _owner;
        _owner = newOwner;
        emit OwnershipTransferred(oldOwner, newOwner);
    }
}
```



```
pragma solidity ^0.8.19;
import {Context} from "../utils/Context.sol";

abstract contract Ownable is Context {
    address private _owner;

    error OwnableUnauthorizedAccount(address account);
    error OwnableInvalidOwner(address owner);
    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

    constructor(address initialOwner) {
        _transferOwnership(initialOwner);
    }

    modifier onlyOwner() {
        _checkOwner();
        _;
    }

    function owner() public view virtual returns (address) {
        return _owner;
    }

    function _checkOwner() internal view virtual {
        if (owner() != _msgSender()) {
            revert OwnableUnauthorizedAccount(_msgSender());
        }
    }

    function renounceOwnership() public virtual onlyOwner {
        _transferOwnership(address(0));
    }

    function transferOwnership(address newOwner) public virtual onlyOwner {
        if (newOwner == address(0)) {
            revert OwnableInvalidOwner(address(0));
        }
        _transferOwnership(newOwner);
    }

    function _transferOwnership(address newOwner) internal virtual {
        address oldOwner = _owner;
        _owner = newOwner;
        emit OwnershipTransferred(oldOwner, newOwner);
    }
}
```

Explicit imports, Custom errors, abi.encodeCall, bytes.concat, string.concat, ...

# More modern solidity

Catching up with the compiler

```
pragma solidity ^0.8.0;
import "../utils/Context.sol";

abstract contract Ownable is Context {
    address private _owner;

    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

    constructor() {
        _transferOwnership(_msgSender());
    }

    modifier onlyOwner() {
        _checkOwner();
        _;
    }

    function owner() public view virtual returns (address) {
        return _owner;
    }

    function _checkOwner() internal view virtual {
        require(owner() == _msgSender(), "Ownable: caller is not the owner");
    }

    function renounceOwnership() public virtual onlyOwner {
        _transferOwnership(address(0));
    }

    function transferOwnership(address newOwner) public virtual onlyOwner {
        require(newOwner != address(0), "Ownable: new owner is the zero address");
        _transferOwnership(newOwner);
    }

    function _transferOwnership(address newOwner) internal virtual {
        address oldOwner = _owner;
        _owner = newOwner;
        emit OwnershipTransferred(oldOwner, newOwner);
    }
}
```

```
pragma solidity ^0.8.19;
import {Context} from "../utils/Context.sol";

abstract contract Ownable is Context {
    address private _owner;

    error OwnableUnauthorizedAccount(address account);
    error OwnableInvalidOwner(address owner);
    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

    constructor(address initialOwner) {
        _transferOwnership(initialOwner);
    }

    modifier onlyOwner() {
        _checkOwner();
        _;
    }

    function owner() public view virtual returns (address) {
        return _owner;
    }

    function _checkOwner() internal view virtual {
        if (owner() != _msgSender()) {
            revert OwnableUnauthorizedAccount(_msgSender());
        }
    }

    function renounceOwnership() public virtual onlyOwner {
        _transferOwnership(address(0));
    }

    function transferOwnership(address newOwner) public virtual onlyOwner {
        if (newOwner == address(0)) {
            revert OwnableInvalidOwner(address(0));
        }
        _transferOwnership(newOwner);
    }

    function _transferOwnership(address newOwner) internal virtual {
        address oldOwner = _owner;
        _owner = newOwner;
        emit OwnershipTransferred(oldOwner, newOwner);
    }
}
```

Explicit imports, Custom errors, abi.encodeCall, bytes.concat, string.concat, ...

# More efficient solidity

Because we do care about gas

```
struct ProposalCore {  
    // --- start retyped from Timers.BlockNumber at offset 0x00 ---  
    uint64 voteStart;  
    address proposer;  
    bytes4 __gap_unused0;  
    // --- start retyped from Timers.BlockNumber at offset 0x20 ---  
    uint64 voteEnd;  
    bytes24 __gap_unused1;  
    // --- Remaining fields starting at offset 0x40 -----  
    bool executed;  
    bool canceled;  
}
```

```
struct ProposalCore {  
    address proposer;  
    uint48 voteStart;  
    uint32 voteDuration;  
    bool executed;  
    bool canceled;  
}
```

Packing storage, using immutable variables, ...

# Removing old code

Some of these might be reintroduced after a redesign (in 5.1 or later)

- **Address.isContract**
- **Checkpoints.History**
- **Counters**
- **SafeMath**
- **SignedSafeMath**
- **Timers**
- **GovernorCompatibilityBravo**
- **GovernorVotesComp**
- **GovernorProposalThreshold**
- **TokenTimelock** (in favor of **VestingWallet**)
- All cross-chain contracts (including **AccessControlCrossChain**)
- All presets in favor of [OpenZeppelin Contracts Wizard](#)
- **ERC20Snapshot**
- **ERC20VotesComp**
- **ERC165Storage**
- **ERC777**
- **ERC1820Implementer**
- **Escrow**
- **ConditionalEscrow**
- **RefundEscrow**
- **PaymentSplitter**
- **PullPayment**

# Some things don't change

We may not like it, but we need it.

```
/**
 * @dev Provides information about the current execution context, including the
 * sender of the transaction and its data. While these are generally available
 * via msg.sender and msg.data, they should not be accessed in such a direct
 * manner, since when dealing with meta-transactions the account sending and
 * paying for execution may not be the actual sender (as far as an application
 * is concerned).
 *
 * This contract is only required for intermediate, library-like contracts.
 */
abstract contract Context {
    function _msgSender() internal view virtual returns (address) {
        return msg.sender;
    }

    function _msgData() internal view virtual returns (bytes calldata) {
        return msg.data;
    }
}
```

**Release candidate coming soon.**

September 1st, 2023



# What to expect after 5.0?

Some things we want to explore, but nothing in this list is guaranteed to happen

## Utilities

- Account abstraction
- Onchain merkle tree construction
- UDVT for common patterns (Masks)

## Governance

- More modules
- More modular design

## DeFi

- New PaymentSplitter

## Nonces (EIP-6077)

- TYPEHASH specific nonce
- Parallel nonces through “tracks”

## Security

- AccessManager extensions
- Circuit breaker

## Upgradeability

- Partial transpilation
- VTable Proxies

## Tests

- Migration to ethers v6
- Migration of FV to CSVL2

## Support the future of ethereum

- Transient storage
- Storage structures for a post-Verge network

**@openzeppelin/contracts**  
**docs.openzeppelin.com**  
**forum.openzeppelin.com**  
**defender.openzeppelin.com**

# Questions?





**OpenZeppelin**

<https://openzeppelin.com>