Namespace Storage

(for now and for a Verkle EVM)

Hadrien Croubois (@Amxx) – Research Engineer @ OpenZeppelin

The grammar of storage layouts

Given the linearized contract ...

... take all the variables in order ...

... and follow the grammar (with root 0)

$$
\begin{aligned}
L_{root} := \quad & root \\
& \mid L_{root} + n \\
& \mid \texttt{keccak256}(L_{root}) \\
& \mid \texttt{keccak256}(H(k) \oplus L_{root}) \\
& \mid \texttt{keccak256}(L_{root} \oplus H(k))
\end{aligned}
$$

Storage matters … when performing upgrades

Upgradeable smart contract needs to preserve a similar view of the storage layout across upgrades.

A lot can potentially go wrong:

- Inserting a variable that shift the layout
- Resize a variable (sometimes it's ok, not always)
- Reorder variables in a contract
- Reorder inheritance

The gap approach

Used in

`@openzeppelin/contracts-upgradeable`

3.x – 4.x

```solidity
/**
 * @dev Context variant with ERC2771 support.
 */
abstract contract ERC2771ContextUpgradeable is Initializable, ContextUpgradeable {
    /// @custom:oz-upgrades-unsafe-allow state-variable-immutable
    address private immutable _trustedForwarder;

    /// @custom:oz-upgrades-unsafe-allow constructor
    constructor(address trustedForwarder) {
        _trustedForwarder = trustedForwarder;
    }

    function isTrustedForwarder(address forwarder) public view virtual returns (bool) {
        return forwarder == _trustedForwarder;
    }

    function _msgSender() internal view virtual override returns (address sender) {
        if (isTrustedForwarder(msg.sender) && msg.data.length >= 20) {
            // The assembly code is more direct than the Solidity version using `abi.decode`.
            /// @solidity memory-safe-assembly
            assembly {
                sender := shr(96, calldataload(sub(calldatasize(), 20)))
            }
        } else {
            return super._msgSender();
        }
    }

    function _msgData() internal view virtual override returns (bytes calldata) {
        if (isTrustedForwarder(msg.sender) && msg.data.length >= 20) {
            return msg.data[:msg.data.length - 20];
        } else {
            return super._msgData();
        }
    }

    /**
     * @dev This empty reserved space is put in place to allow future versions to add new
     * variables without shifting down storage in the inheritance chain.
     * See https://docs.openzeppelin.com/contracts/4.x/upgradeable#storage_gaps
     */
    uint256[50] private __gap;
}
```

The namespace approach

Used in

`@openzeppelin/contracts-upgradeable`

5.x

```solidity
abstract contract OwnableUpgradeable is Initializable, ContextUpgradeable {
    /// @custom:storage-location erc7201:openzeppelin.storage.Ownable
    struct OwnableStorage {
        address _owner;
    }

    // keccak256(abi.encode(uint256(keccak256("openzeppelin.storage.Ownable")) - 1)) & ~bytes32(uint256(0xff))
    bytes32 private constant OwnableStorageLocation = 0x9016d09d72d40fdae2fd8ceac6b6234c7706214fd39c1cd1e609a0528c199300;

    function _getOwnableStorage() private pure returns (OwnableStorage storage $) {
        assembly {
            $.slot := OwnableStorageLocation
        }
    }

    /**
     * @dev The caller account is not authorized to perform an operation.
     */
    error OwnableUnauthorizedAccount(address account);

    /**
     * @dev The owner is not a valid owner account. (eg. `address(0)`)
     */
    error OwnableInvalidOwner(address owner);

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

    /**
     * @dev Initializes the contract setting the address provided by the deployer as the initial owner.
     */
    function __Ownable_init(address initialOwner) internal onlyInitializing {
        __Ownable_init_unchained(initialOwner);
    }

    function __Ownable_init_unchained(address initialOwner) internal onlyInitializing {
        if (initialOwner == address(0)) {
            revert OwnableInvalidOwner(address(0));
        }
        _transferOwnership(initialOwner);
    }

    /**
     * @dev Throws if called by any account other than the owner.
     */
    modifier onlyOwner() {
        _checkOwner();
        _;
    }

    /**
     * @dev Returns the address of the current owner.
     */
    function owner() public view virtual returns (address) {
        OwnableStorage storage $ = _getOwnableStorage();
        return $._owner;
    }
```

ERC-7201

ERC-7201: derivation of a new (safe) root

```
21    abstract contract OwnableUpgradeable is Initializable, ContextUpgradeable {
22        /// @custom:storage-location erc7201:openzeppelin.storage.Ownable
23        struct OwnableStorage {
24            address _owner;
25        }
26
27        // keccak256(abi.encode(uint256(keccak256("openzeppelin.storage.Ownable")) - 1)) & ~bytes32(uint256(0xff))
28        bytes32 private constant OwnableStorageLocation = 0x9016d09d72d40fdae2fd8ceac6b6234c7706214fd39c1cd1e609a0528c199300;
29
30        function _getOwnableStorage() private pure returns (OwnableStorage storage $) {
31            assembly {
32                $.slot := OwnableStorageLocation
33            }
34        }
```

Going back to the grammar

What is not in the grammar for a root 0 ?

$$
\begin{aligned}
L_{root} := \quad & root \\
& \mid L_{root} + n \\
& \mid \texttt{keccak256}(L_{root}) \\
& \mid \texttt{keccak256}(H(k) \oplus L_{root}) \\
& \mid \texttt{keccak256}(L_{root} \oplus H(k))
\end{aligned}
$$

If *n* is small enough, and if *keccak256* is a good hashing function, we just need to find a location that is not in L_{root}, and use that as our new root.

ERC-7201 derivation explained

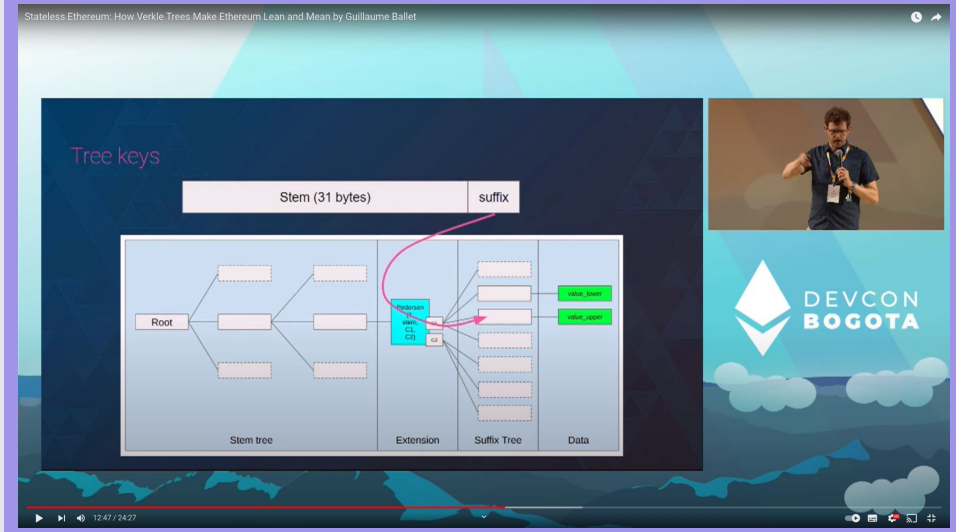
```
keccak256(keccak256("openzeppelin.storage.Ownable") - 1) & ~0xFF
```

`"openzeppelin.storage.Ownable"`	Seed
`keccak256(...) - 1`	Hash of the seed might collide if the seed matches a valid storage location is L_0, so we subtract 1 to exit L_0
`keccak256(...)`	Get a completely new space (so doing "+n") don't bring us back into L0
`... & ~0xFF`	Clean the last byte so our namespace starts at the beginning of a bucket

What is a bucket?

Should Solidity (& Vyper) change their grammar?

Maybe!



Stateless Ethereum: How Verkle Trees Make Ethereum Lean and Mean by Guillaume Ballet
https://www.youtube.com/watch?v=Q7rStTKwuYs

Thursday,
Nov 16, 2023

Istanbul, Türkiye

Welcome to the Solidity Summit template!

First of all, thank you so much for being a speaker & sharing your knowledge with the audience! 💙

Please help us make Solidity Summit 2023 a high-quality and enjoyable conference experience for all by…

- Refraining from marketing-focused talks or pure product presentations
- Keeping the content technical, informational and educational
- Asking yourself: If you were in the audience, would you find your talk interesting, helpful, and relevant?

Looking forward to seeing your talk at the summit and don't hesitate to reach out to us with any questions!

Of course you don’t *have* to use the template provided!

You can also use your own slide design by simply deleting all these slides and copy pasting in an existing presentation or creating your own from scratch!

There is a dark mode

…and a light mode

*(at the end of this presentation)

ADD YOUR TITLE HERE

SUBTITLE AND/OR SPEAKER INFO

This is a regular slide.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer eget volutpat eros, sit amet tristique enim. Aliquam nec odio eu ligula aliquam ultrices. Nulla fermentum dui nisl, ut tempus eros volutpat convallis.

- Bullet lists
- Content
- More content

1. Numbered lists
    a. Etc.
    b. Also
    c. Moreover,

This is a slide with two columns.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer eget volutpat eros, sit amet tristique enim. Aliquam nec odio eu ligula aliquam ultrices. Nulla fermentum dui nisl, ut tempus eros volutpat convallis.

- Point 1
- Point 2
- Point 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer eget volutpat eros, sit amet tristique enim. Aliquam nec odio eu ligula aliquam ultrices. Nulla fermentum dui nisl, ut tempus eros volutpat convallis.

- Point 1
- Point 2
- Point 3

This is a split slide.

Space for subtitles.

Add text or a visualization here.

You can also use title only slides and add content of your choice!

Thursday,
Nov 16, 2023

Istanbul, Türkiye

ADD YOUR TITLE HERE

SUBTITLE AND/OR SPEAKER INFO

This is a regular slide.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer eget volutpat eros, sit amet tristique enim. Aliquam nec odio eu ligula aliquam ultrices. Nulla fermentum dui nisl, ut tempus eros volutpat convallis.

- Bullet lists
- Content
- More content

1. Numbered lists
    a. Etc.
    b. Also
    c. Moreover,

This is a slide with two columns.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer eget volutpat eros, sit amet tristique enim. Aliquam nec odio eu ligula aliquam ultrices. Nulla fermentum dui nisl, ut tempus eros volutpat convallis.

- Point 1
- Point 2
- Point 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer eget volutpat eros, sit amet tristique enim. Aliquam nec odio eu ligula aliquam ultrices. Nulla fermentum dui nisl, ut tempus eros volutpat convallis.

- Point 1
- Point 2
- Point 3

This is a split slide.

Space for subtitles.

Add text or a visualization here.

You can also use title only slides and add content of your choice!

Thursday,
Nov 16, 2023

Istanbul, Türkiye