Apr 8, 2023 1 min read

# Openzeppelin Ownable: Use Ownable2Step Instead

**Updated: May 16, 2023**

**The onlyOwner modifier is probably one of the most common patterns in Solidity.**

**In the following example, the function setMessage can only be called by the address designated to be the owner.**

```
function setMessage(string calldata _message) external onlyOwner {
        message = _message;
}
```

**However, the commonly used Openzeppelin ownable implementation has a shortcoming that it allows the owner to transfer ownership to a non-existent or mistyped address.**

**Ownable2Step is safer than Ownable for smart contracts because the owner cannot accidentally transfer smart contract ownership to a mistyped address. Rather than directly transferring to the new owner, the transfer only completes when the new owner accepts ownership.**

**Ownable2Step was released in January 2023 during the OpenZeppelin version 4.8 update. Here are the docs and the code.**

**Here is an minimal example:**

```
import "@openzeppelin/contracts/access/Ownable2Step.sol";

contract ExampleOwnable2Step is Ownable2Step {
    message public string;

    function setMessage(string calldata _message) external onlyOwner {
        message = _message;
    }
}
```

**Note that little changes from the normal Ownable implementation except that a different library is imported and the contract inherits from Ownable2Step instead of Ownable.**

**Ownable2Step inherits from Ownable and overrides "transferOwnership()" to make the new owner "pending." The receiver must then call "acceptOwnership()" to finalize the transfer. This ensures only**

**an address that has access to it's private keys, or control of the smart contract address, can control the smart contract.**

**There is still no two-step verification for renouncing ownership, i.e. transferring ownership to the zero address. If there is no need to renounce ownership, then it is safer to override "renounceOwnership()" to revert when called.**

# Learn More

**This tutorial is part of our advanced solidity training course.**