**Proxies**

This is a low-level set of contracts implementing different proxy patterns with and without upgradeability. For an in-depth overview of this pattern check out the [Proxy Upgrade Pattern](https://docs.openzeppelin.com/upgrades-plugins/1.x/proxies) page.

Most of the proxies below are built on an abstract base contract.

* [Proxy](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy): Abstract contract implementing the core delegation functionality.

In order to avoid clashes with the storage variables of the implementation contract behind a proxy, we use [EIP1967](https://eips.ethereum.org/EIPS/eip-1967) storage slots.

* [ERC1967Upgrade](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade): Internal functions to get and set the storage slots defined in EIP1967.
* [ERC1967Proxy](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Proxy): A proxy using EIP1967 storage slots. Not upgradeable by default.

There are two alternative ways to add upgradeability to an ERC1967 proxy. Their differences are explained below in [Transparent vs UUPS Proxies](https://docs.openzeppelin.com/contracts/4.x/api/proxy#transparent-vs-uups).

* [TransparentUpgradeableProxy](https://docs.openzeppelin.com/contracts/4.x/api/proxy#TransparentUpgradeableProxy): A proxy with a built in admin and upgrade interface.
* [UUPSUpgradeable](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UUPSUpgradeable): An upgradeability mechanism to be included in the implementation contract.

|  |  |
| --- | --- |
|  | Using upgradeable proxies correctly and securely is a difficult task that requires deep knowledge of the proxy pattern, Solidity, and the EVM. Unless you want a lot of low level control, we recommend using the [OpenZeppelin Upgrades Plugins](https://docs.openzeppelin.com/upgrades-plugins/1.x/) for Truffle and Hardhat. |

A different family of proxies are beacon proxies. This pattern, popularized by Dharma, allows multiple proxies to be upgraded to a different implementation in a single transaction.

* [BeaconProxy](https://docs.openzeppelin.com/contracts/4.x/api/proxy#BeaconProxy): A proxy that retrieves its implementation from a beacon contract.
* [UpgradeableBeacon](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UpgradeableBeacon): A beacon contract with a built in admin that can upgrade the [BeaconProxy](https://docs.openzeppelin.com/contracts/4.x/api/proxy#BeaconProxy) pointing to it.

In this pattern, the proxy contract doesn’t hold the implementation address in storage like an ERC1967 proxy. Instead, the address is stored in a separate beacon contract. The upgrade operations are sent to the beacon instead of to the proxy contract, and all proxies that follow that beacon are automatically upgraded.

Outside the realm of upgradeability, proxies can also be useful to make cheap contract clones, such as those created by an on-chain factory contract that creates many instances of the same contract. These instances are designed to be both cheap to deploy, and cheap to call.

* [Clones](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Clones): A library that can deploy cheap minimal non-upgradeable proxies.

**Transparent vs UUPS Proxies**

The original proxies included in OpenZeppelin followed the [Transparent Proxy Pattern](https://blog.openzeppelin.com/the-transparent-proxy-pattern/). While this pattern is still provided, our recommendation is now shifting towards UUPS proxies, which are both lightweight and versatile. The name UUPS comes from [EIP1822](https://eips.ethereum.org/EIPS/eip-1822), which first documented the pattern.

While both of these share the same interface for upgrades, in UUPS proxies the upgrade is handled by the implementation, and can eventually be removed. Transparent proxies, on the other hand, include the upgrade and admin logic in the proxy itself. This means [TransparentUpgradeableProxy](https://docs.openzeppelin.com/contracts/4.x/api/proxy#TransparentUpgradeableProxy) is more expensive to deploy than what is possible with UUPS proxies.

UUPS proxies are implemented using an [ERC1967Proxy](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Proxy). Note that this proxy is not by itself upgradeable. It is the role of the implementation to include, alongside the contract’s logic, all the code necessary to update the implementation’s address that is stored at a specific slot in the proxy’s storage space. This is where the [UUPSUpgradeable](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UUPSUpgradeable) contract comes in. Inheriting from it (and overriding the [\_authorizeUpgrade](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UUPSUpgradeable-_authorizeUpgrade-address-) function with the relevant access control mechanism) will turn your contract into a UUPS compliant implementation.

Note that since both proxies use the same storage slot for the implementation address, using a UUPS compliant implementation with a [TransparentUpgradeableProxy](https://docs.openzeppelin.com/contracts/4.x/api/proxy#TransparentUpgradeableProxy) might allow non-admins to perform upgrade operations.

By default, the upgrade functionality included in [UUPSUpgradeable](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UUPSUpgradeable) contains a security mechanism that will prevent any upgrades to a non UUPS compliant implementation. This prevents upgrades to an implementation contract that wouldn’t contain the necessary upgrade mechanism, as it would lock the upgradeability of the proxy forever. This security mechanism can be bypassed by either of:

* Adding a flag mechanism in the implementation that will disable the upgrade function when triggered.
* Upgrading to an implementation that features an upgrade mechanism without the additional security check, and then upgrading again to another implementation without the upgrade mechanism.

The current implementation of this security mechanism uses [EIP1822](https://eips.ethereum.org/EIPS/eip-1822) to detect the storage slot used by the implementation. A previous implementation, now deprecated, relied on a rollback check. It is possible to upgrade from a contract using the old mechanism to a new one. The inverse is however not possible, as old implementations (before version 4.5) did not include the ERC1822 interface.

**Core**

**Proxy**

import "@openzeppelin/contracts/proxy/Proxy.sol";

This abstract contract provides a fallback function that delegates all calls to another contract using the EVM instruction delegatecall. We refer to the second contract as the *implementation* behind the proxy, and it has to be specified by overriding the virtual [\_implementation](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-_implementation--) function.

Additionally, delegation to the implementation can be triggered manually through the [\_fallback](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-_fallback--) function, or to a different contract through the [\_delegate](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-_delegate-address-) function.

The success and return data of the delegated call will be returned back to the caller of the proxy.

**FUNCTIONS**

* [\_delegate(implementation)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-_delegate-address-)
* [\_implementation()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-_implementation--)
* [\_fallback()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-_fallback--)
* [fallback()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-fallback--)
* [receive()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-receive--)
* [\_beforeFallback()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-_beforeFallback--)

**\_delegate(address implementation)**internal

Delegates the current call to implementation.

This function does not return to its internal call site, it will return directly to the external caller.

**\_implementation() → address**internal

This is a virtual function that should be overridden so it returns the address to which the fallback function and [\_fallback](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-_fallback--) should delegate.

**\_fallback()**internal

Delegates the current call to the address returned by \_implementation().

This function does not return to its internal call site, it will return directly to the external caller.

**fallback()**external

Fallback function that delegates calls to the address returned by \_implementation(). Will run if no other function in the contract matches the call data.

**receive()**external

Fallback function that delegates calls to the address returned by \_implementation(). Will run if call data is empty.

**\_beforeFallback()**internal

Hook that is called before falling back to the implementation. Can happen as part of a manual \_fallback call, or as part of the Solidity fallback or receive functions.

If overridden should call super.\_beforeFallback().

**ERC1967**

**ERC1967Proxy**

import "@openzeppelin/contracts/proxy/ERC1967/ERC1967Proxy.sol";

This contract implements an upgradeable proxy. It is upgradeable because calls are delegated to an implementation address that can be changed. This address is stored in storage in the location specified by [EIP1967](https://eips.ethereum.org/EIPS/eip-1967), so that it doesn’t conflict with the storage layout of the implementation behind the proxy.

**FUNCTIONS**

* [constructor(\_logic, \_data)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Proxy-constructor-address-bytes-)
* [\_implementation()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Proxy-_implementation--)

**ERC1967UPGRADE**

* [\_getImplementation()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_getImplementation--)
* [\_upgradeTo(newImplementation)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeTo-address-)
* [\_upgradeToAndCall(newImplementation, data, forceCall)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeToAndCall-address-bytes-bool-)
* [\_upgradeToAndCallUUPS(newImplementation, data, forceCall)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeToAndCallUUPS-address-bytes-bool-)
* [\_getAdmin()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_getAdmin--)
* [\_changeAdmin(newAdmin)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_changeAdmin-address-)
* [\_getBeacon()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_getBeacon--)
* [\_upgradeBeaconToAndCall(newBeacon, data, forceCall)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeBeaconToAndCall-address-bytes-bool-)

**PROXY**

* [\_delegate(implementation)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-_delegate-address-)
* [\_fallback()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-_fallback--)
* [fallback()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-fallback--)
* [receive()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-receive--)
* [\_beforeFallback()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-_beforeFallback--)

**EVENTS**

**IERC1967**

* [Upgraded(implementation)](https://docs.openzeppelin.com/contracts/4.x/api/interfaces#IERC1967-Upgraded-address-)
* [AdminChanged(previousAdmin, newAdmin)](https://docs.openzeppelin.com/contracts/4.x/api/interfaces#IERC1967-AdminChanged-address-address-)
* [BeaconUpgraded(beacon)](https://docs.openzeppelin.com/contracts/4.x/api/interfaces#IERC1967-BeaconUpgraded-address-)

**constructor(address \_logic, bytes \_data)**public

Initializes the upgradeable proxy with an initial implementation specified by \_logic.

If \_data is nonempty, it’s used as data in a delegate call to \_logic. This will typically be an encoded function call, and allows initializing the storage of the proxy like a Solidity constructor.

**\_implementation() → address impl**internal

Returns the current implementation address.

**ERC1967Upgrade**

import "@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol";

This abstract contract provides getters and event emitting update functions for [EIP1967](https://eips.ethereum.org/EIPS/eip-1967) slots.

*Available since v4.1.*

**FUNCTIONS**

* [\_getImplementation()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_getImplementation--)
* [\_upgradeTo(newImplementation)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeTo-address-)
* [\_upgradeToAndCall(newImplementation, data, forceCall)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeToAndCall-address-bytes-bool-)
* [\_upgradeToAndCallUUPS(newImplementation, data, forceCall)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeToAndCallUUPS-address-bytes-bool-)
* [\_getAdmin()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_getAdmin--)
* [\_changeAdmin(newAdmin)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_changeAdmin-address-)
* [\_getBeacon()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_getBeacon--)
* [\_upgradeBeaconToAndCall(newBeacon, data, forceCall)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeBeaconToAndCall-address-bytes-bool-)

**EVENTS**

**IERC1967**

* [Upgraded(implementation)](https://docs.openzeppelin.com/contracts/4.x/api/interfaces#IERC1967-Upgraded-address-)
* [AdminChanged(previousAdmin, newAdmin)](https://docs.openzeppelin.com/contracts/4.x/api/interfaces#IERC1967-AdminChanged-address-address-)
* [BeaconUpgraded(beacon)](https://docs.openzeppelin.com/contracts/4.x/api/interfaces#IERC1967-BeaconUpgraded-address-)

**\_getImplementation() → address**internal

Returns the current implementation address.

**\_upgradeTo(address newImplementation)**internal

Perform implementation upgrade

Emits an {Upgraded} event.

**\_upgradeToAndCall(address newImplementation, bytes data, bool forceCall)**internal

Perform implementation upgrade with additional setup call.

Emits an {Upgraded} event.

**\_upgradeToAndCallUUPS(address newImplementation, bytes data, bool forceCall)**internal

Perform implementation upgrade with security checks for UUPS proxies, and additional setup call.

Emits an {Upgraded} event.

**\_getAdmin() → address**internal

Returns the current admin.

**\_changeAdmin(address newAdmin)**internal

Changes the admin of the proxy.

Emits an {AdminChanged} event.

**\_getBeacon() → address**internal

Returns the current beacon.

**\_upgradeBeaconToAndCall(address newBeacon, bytes data, bool forceCall)**internal

Perform beacon upgrade with additional setup call. Note: This upgrades the address of the beacon, it does not upgrade the implementation contained in the beacon (see [UpgradeableBeacon.\_setImplementation](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UpgradeableBeacon-_setImplementation-address-) for that).

Emits a {BeaconUpgraded} event.

**Transparent Proxy**

**TransparentUpgradeableProxy**

import "@openzeppelin/contracts/proxy/transparent/TransparentUpgradeableProxy.sol";

This contract implements a proxy that is upgradeable by an admin.

To avoid [proxy selector clashing](https://medium.com/nomic-labs-blog/malicious-backdoors-in-ethereum-proxies-62629adf3357), which can potentially be used in an attack, this contract uses the [transparent proxy pattern](https://blog.openzeppelin.com/the-transparent-proxy-pattern/). This pattern implies two things that go hand in hand:

1. If any account other than the admin calls the proxy, the call will be forwarded to the implementation, even if that call matches one of the admin functions exposed by the proxy itself.
2. If the admin calls the proxy, it can access the admin functions, but its calls will never be forwarded to the implementation. If the admin tries to call a function on the implementation it will fail with an error that says "admin cannot fallback to proxy target".

These properties mean that the admin account can only be used for admin actions like upgrading the proxy or changing the admin, so it’s best if it’s a dedicated account that is not used for anything else. This will avoid headaches due to sudden errors when trying to call a function from the proxy implementation.

Our recommendation is for the dedicated account to be an instance of the [ProxyAdmin](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ProxyAdmin) contract. If set up this way, you should think of the ProxyAdmin instance as the real administrative interface of your proxy.

|  |  |
| --- | --- |
|  | The real interface of this proxy is that defined in ITransparentUpgradeableProxy. This contract does not inherit from that interface, and instead the admin functions are implicitly implemented using a custom dispatch mechanism in \_fallback. Consequently, the compiler will not produce an ABI for this contract. This is necessary to fully implement transparency without decoding reverts caused by selector clashes between the proxy and the implementation. |
|  | It is not recommended to extend this contract to add additional external functions. If you do so, the compiler will not check that there are no selector conflicts, due to the note above. A selector clash between any new function and the functions declared in [ITransparentUpgradeableProxy](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ITransparentUpgradeableProxy) will be resolved in favor of the new one. This could render the admin operations inaccessible, which could prevent upgradeability. Transparency may also be compromised. |

**MODIFIERS**

* [ifAdmin()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#TransparentUpgradeableProxy-ifAdmin--)

**FUNCTIONS**

* [constructor(\_logic, admin\_, \_data)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#TransparentUpgradeableProxy-constructor-address-address-bytes-)
* [\_fallback()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#TransparentUpgradeableProxy-_fallback--)
* [\_admin()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#TransparentUpgradeableProxy-_admin--)

**ERC1967PROXY**

* [\_implementation()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Proxy-_implementation--)

**ERC1967UPGRADE**

* [\_getImplementation()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_getImplementation--)
* [\_upgradeTo(newImplementation)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeTo-address-)
* [\_upgradeToAndCall(newImplementation, data, forceCall)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeToAndCall-address-bytes-bool-)
* [\_upgradeToAndCallUUPS(newImplementation, data, forceCall)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeToAndCallUUPS-address-bytes-bool-)
* [\_getAdmin()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_getAdmin--)
* [\_changeAdmin(newAdmin)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_changeAdmin-address-)
* [\_getBeacon()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_getBeacon--)
* [\_upgradeBeaconToAndCall(newBeacon, data, forceCall)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeBeaconToAndCall-address-bytes-bool-)

**PROXY**

* [\_delegate(implementation)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-_delegate-address-)
* [fallback()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-fallback--)
* [receive()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-receive--)
* [\_beforeFallback()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-_beforeFallback--)

**EVENTS**

**IERC1967**

* [Upgraded(implementation)](https://docs.openzeppelin.com/contracts/4.x/api/interfaces#IERC1967-Upgraded-address-)
* [AdminChanged(previousAdmin, newAdmin)](https://docs.openzeppelin.com/contracts/4.x/api/interfaces#IERC1967-AdminChanged-address-address-)
* [BeaconUpgraded(beacon)](https://docs.openzeppelin.com/contracts/4.x/api/interfaces#IERC1967-BeaconUpgraded-address-)

**ifAdmin()**modifier

Modifier used internally that will delegate the call to the implementation unless the sender is the admin.

|  |  |
| --- | --- |
|  | This modifier is deprecated, as it could cause issues if the modified function has arguments, and the implementation provides a function with the same selector. |

**constructor(address \_logic, address admin\_, bytes \_data)**public

Initializes an upgradeable proxy managed by \_admin, backed by the implementation at \_logic, and optionally initialized with \_data as explained in [ERC1967Proxy.constructor](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Proxy-constructor-address-bytes-).

**\_fallback()**internal

If caller is the admin process the call internally, otherwise transparently fallback to the proxy behavior

**\_admin() → address**internal

Returns the current admin.

**ProxyAdmin**

import "@openzeppelin/contracts/proxy/transparent/ProxyAdmin.sol";

This is an auxiliary contract meant to be assigned as the admin of a [TransparentUpgradeableProxy](https://docs.openzeppelin.com/contracts/4.x/api/proxy#TransparentUpgradeableProxy). For an explanation of why you would want to use this see the documentation for [TransparentUpgradeableProxy](https://docs.openzeppelin.com/contracts/4.x/api/proxy#TransparentUpgradeableProxy).

**FUNCTIONS**

* [getProxyImplementation(proxy)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ProxyAdmin-getProxyImplementation-contract-ITransparentUpgradeableProxy-)
* [getProxyAdmin(proxy)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ProxyAdmin-getProxyAdmin-contract-ITransparentUpgradeableProxy-)
* [changeProxyAdmin(proxy, newAdmin)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ProxyAdmin-changeProxyAdmin-contract-ITransparentUpgradeableProxy-address-)
* [upgrade(proxy, implementation)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ProxyAdmin-upgrade-contract-ITransparentUpgradeableProxy-address-)
* [upgradeAndCall(proxy, implementation, data)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ProxyAdmin-upgradeAndCall-contract-ITransparentUpgradeableProxy-address-bytes-)

**OWNABLE**

* [owner()](https://docs.openzeppelin.com/contracts/4.x/api/access#Ownable-owner--)
* [\_checkOwner()](https://docs.openzeppelin.com/contracts/4.x/api/access#Ownable-_checkOwner--)
* [renounceOwnership()](https://docs.openzeppelin.com/contracts/4.x/api/access#Ownable-renounceOwnership--)
* [transferOwnership(newOwner)](https://docs.openzeppelin.com/contracts/4.x/api/access#Ownable-transferOwnership-address-)
* [\_transferOwnership(newOwner)](https://docs.openzeppelin.com/contracts/4.x/api/access#Ownable-_transferOwnership-address-)

**EVENTS**

**OWNABLE**

* [OwnershipTransferred(previousOwner, newOwner)](https://docs.openzeppelin.com/contracts/4.x/api/access#Ownable-OwnershipTransferred-address-address-)

**getProxyImplementation(contract ITransparentUpgradeableProxy proxy) → address**public

Returns the current implementation of proxy.

Requirements:

* This contract must be the admin of proxy.

**getProxyAdmin(contract ITransparentUpgradeableProxy proxy) → address**public

Returns the current admin of proxy.

Requirements:

* This contract must be the admin of proxy.

**changeProxyAdmin(contract ITransparentUpgradeableProxy proxy, address newAdmin)**public

Changes the admin of proxy to newAdmin.

Requirements:

* This contract must be the current admin of proxy.

**upgrade(contract ITransparentUpgradeableProxy proxy, address implementation)**public

Upgrades proxy to implementation. See {TransparentUpgradeableProxy-upgradeTo}.

Requirements:

* This contract must be the admin of proxy.

**upgradeAndCall(contract ITransparentUpgradeableProxy proxy, address implementation, bytes data)**public

Upgrades proxy to implementation and calls a function on the new implementation. See {TransparentUpgradeableProxy-upgradeToAndCall}.

Requirements:

* This contract must be the admin of proxy.

**Beacon**

**BeaconProxy**

import "@openzeppelin/contracts/proxy/beacon/BeaconProxy.sol";

This contract implements a proxy that gets the implementation address for each call from an [UpgradeableBeacon](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UpgradeableBeacon).

The beacon address is stored in storage slot uint256(keccak256('eip1967.proxy.beacon')) - 1, so that it doesn’t conflict with the storage layout of the implementation behind the proxy.

*Available since v3.4.*

**FUNCTIONS**

* [constructor(beacon, data)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#BeaconProxy-constructor-address-bytes-)
* [\_beacon()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#BeaconProxy-_beacon--)
* [\_implementation()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#BeaconProxy-_implementation--)
* [\_setBeacon(beacon, data)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#BeaconProxy-_setBeacon-address-bytes-)

**ERC1967UPGRADE**

* [\_getImplementation()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_getImplementation--)
* [\_upgradeTo(newImplementation)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeTo-address-)
* [\_upgradeToAndCall(newImplementation, data, forceCall)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeToAndCall-address-bytes-bool-)
* [\_upgradeToAndCallUUPS(newImplementation, data, forceCall)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeToAndCallUUPS-address-bytes-bool-)
* [\_getAdmin()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_getAdmin--)
* [\_changeAdmin(newAdmin)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_changeAdmin-address-)
* [\_getBeacon()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_getBeacon--)
* [\_upgradeBeaconToAndCall(newBeacon, data, forceCall)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeBeaconToAndCall-address-bytes-bool-)

**PROXY**

* [\_delegate(implementation)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-_delegate-address-)
* [\_fallback()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-_fallback--)
* [fallback()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-fallback--)
* [receive()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-receive--)
* [\_beforeFallback()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Proxy-_beforeFallback--)

**EVENTS**

**IERC1967**

* [Upgraded(implementation)](https://docs.openzeppelin.com/contracts/4.x/api/interfaces#IERC1967-Upgraded-address-)
* [AdminChanged(previousAdmin, newAdmin)](https://docs.openzeppelin.com/contracts/4.x/api/interfaces#IERC1967-AdminChanged-address-address-)
* [BeaconUpgraded(beacon)](https://docs.openzeppelin.com/contracts/4.x/api/interfaces#IERC1967-BeaconUpgraded-address-)

**constructor(address beacon, bytes data)**public

Initializes the proxy with beacon.

If data is nonempty, it’s used as data in a delegate call to the implementation returned by the beacon. This will typically be an encoded function call, and allows initializing the storage of the proxy like a Solidity constructor.

Requirements:

* beacon must be a contract with the interface [IBeacon](https://docs.openzeppelin.com/contracts/4.x/api/proxy#IBeacon).

**\_beacon() → address**internal

Returns the current beacon address.

**\_implementation() → address**internal

Returns the current implementation address of the associated beacon.

**\_setBeacon(address beacon, bytes data)**internal

Changes the proxy to use a new beacon. Deprecated: see [\_upgradeBeaconToAndCall](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeBeaconToAndCall-address-bytes-bool-).

If data is nonempty, it’s used as data in a delegate call to the implementation returned by the beacon.

Requirements:

* beacon must be a contract.
* The implementation returned by beacon must be a contract.

**IBeacon**

import "@openzeppelin/contracts/proxy/beacon/IBeacon.sol";

This is the interface that [BeaconProxy](https://docs.openzeppelin.com/contracts/4.x/api/proxy#BeaconProxy) expects of its beacon.

**FUNCTIONS**

* [implementation()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#IBeacon-implementation--)

**implementation() → address**external

Must return an address that can be used as a delegate call target.

[BeaconProxy](https://docs.openzeppelin.com/contracts/4.x/api/proxy#BeaconProxy) will check that this address is a contract.

**UpgradeableBeacon**

import "@openzeppelin/contracts/proxy/beacon/UpgradeableBeacon.sol";

This contract is used in conjunction with one or more instances of [BeaconProxy](https://docs.openzeppelin.com/contracts/4.x/api/proxy#BeaconProxy) to determine their implementation contract, which is where they will delegate all function calls.

An owner is able to change the implementation the beacon points to, thus upgrading the proxies that use this beacon.

**FUNCTIONS**

* [constructor(implementation\_)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UpgradeableBeacon-constructor-address-)
* [implementation()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UpgradeableBeacon-implementation--)
* [upgradeTo(newImplementation)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UpgradeableBeacon-upgradeTo-address-)

**OWNABLE**

* [owner()](https://docs.openzeppelin.com/contracts/4.x/api/access#Ownable-owner--)
* [\_checkOwner()](https://docs.openzeppelin.com/contracts/4.x/api/access#Ownable-_checkOwner--)
* [renounceOwnership()](https://docs.openzeppelin.com/contracts/4.x/api/access#Ownable-renounceOwnership--)
* [transferOwnership(newOwner)](https://docs.openzeppelin.com/contracts/4.x/api/access#Ownable-transferOwnership-address-)
* [\_transferOwnership(newOwner)](https://docs.openzeppelin.com/contracts/4.x/api/access#Ownable-_transferOwnership-address-)

**EVENTS**

* [Upgraded(implementation)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UpgradeableBeacon-Upgraded-address-)

**OWNABLE**

* [OwnershipTransferred(previousOwner, newOwner)](https://docs.openzeppelin.com/contracts/4.x/api/access#Ownable-OwnershipTransferred-address-address-)

**constructor(address implementation\_)**public

Sets the address of the initial implementation, and the deployer account as the owner who can upgrade the beacon.

**implementation() → address**public

Returns the current implementation address.

**upgradeTo(address newImplementation)**public

Upgrades the beacon to a new implementation.

Emits an [Upgraded](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UpgradeableBeacon-Upgraded-address-) event.

Requirements:

* msg.sender must be the owner of the contract.
* newImplementation must be a contract.

**Upgraded(address indexed implementation)**event

Emitted when the implementation returned by the beacon is changed.

**Minimal Clones**

**Clones**

import "@openzeppelin/contracts/proxy/Clones.sol";

[EIP 1167](https://eips.ethereum.org/EIPS/eip-1167) is a standard for deploying minimal proxy contracts, also known as "clones".

To simply and cheaply clone contract functionality in an immutable way, this standard specifies a minimal bytecode implementation that delegates all calls to a known, fixed address.

The library includes functions to deploy a proxy using either create (traditional deployment) or create2 (salted deterministic deployment). It also includes functions to predict the addresses of clones deployed using the deterministic method.

*Available since v3.4.*

**FUNCTIONS**

* [clone(implementation)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Clones-clone-address-)
* [cloneDeterministic(implementation, salt)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Clones-cloneDeterministic-address-bytes32-)
* [predictDeterministicAddress(implementation, salt, deployer)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Clones-predictDeterministicAddress-address-bytes32-address-)
* [predictDeterministicAddress(implementation, salt)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Clones-predictDeterministicAddress-address-bytes32-)

**clone(address implementation) → address instance**internal

Deploys and returns the address of a clone that mimics the behaviour of implementation.

This function uses the create opcode, which should never revert.

**cloneDeterministic(address implementation, bytes32 salt) → address instance**internal

Deploys and returns the address of a clone that mimics the behaviour of implementation.

This function uses the create2 opcode and a salt to deterministically deploy the clone. Using the same implementation and salt multiple time will revert, since the clones cannot be deployed twice at the same address.

**predictDeterministicAddress(address implementation, bytes32 salt, address deployer) → address predicted**internal

Computes the address of a clone deployed using [Clones.cloneDeterministic](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Clones-cloneDeterministic-address-bytes32-).

**predictDeterministicAddress(address implementation, bytes32 salt) → address predicted**internal

Computes the address of a clone deployed using [Clones.cloneDeterministic](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Clones-cloneDeterministic-address-bytes32-).

**Utils**

**Initializable**

import "@openzeppelin/contracts/proxy/utils/Initializable.sol";

This is a base contract to aid in writing upgradeable contracts, or any kind of contract that will be deployed behind a proxy. Since proxied contracts do not make use of a constructor, it’s common to move constructor logic to an external initializer function, usually called initialize. It then becomes necessary to protect this initializer function so it can only be called once. The [initializer](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Initializable-initializer--) modifier provided by this contract will have this effect.

The initialization functions use a version number. Once a version number is used, it is consumed and cannot be reused. This mechanism prevents re-execution of each "step" but allows the creation of new initialization steps in case an upgrade adds a module that needs to be initialized.

For example:

contract MyToken is ERC20Upgradeable {

function initialize() initializer public {

\_\_ERC20\_init("MyToken", "MTK");

}

}

contract MyTokenV2 is MyToken, ERC20PermitUpgradeable {

function initializeV2() reinitializer(2) public {

\_\_ERC20Permit\_init("MyToken");

}

}

|  |  |
| --- | --- |
|  | To avoid leaving the proxy in an uninitialized state, the initializer function should be called as early as possible by providing the encoded function call as the \_data argument to [ERC1967Proxy.constructor](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Proxy-constructor-address-bytes-). |
|  | When used with inheritance, manual care must be taken to not invoke a parent initializer twice, or to ensure that all initializers are idempotent. This is not verified automatically as constructors are by Solidity. |

|  |  |
| --- | --- |
|  | Avoid leaving a contract uninitialized.  An uninitialized contract can be taken over by an attacker. This applies to both a proxy and its implementation contract, which may impact the proxy. To prevent the implementation contract from being used, you should invoke the [\_disableInitializers](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Initializable-_disableInitializers--) function in the constructor to automatically lock it when it is deployed:  /// @custom:oz-upgrades-unsafe-allow constructor  constructor() {  \_disableInitializers();  } |

**MODIFIERS**

* [initializer()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Initializable-initializer--)
* [reinitializer(version)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Initializable-reinitializer-uint8-)
* [onlyInitializing()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Initializable-onlyInitializing--)

**FUNCTIONS**

* [\_disableInitializers()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Initializable-_disableInitializers--)
* [\_getInitializedVersion()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Initializable-_getInitializedVersion--)
* [\_isInitializing()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Initializable-_isInitializing--)

**EVENTS**

* [Initialized(version)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Initializable-Initialized-uint8-)

**initializer()**modifier

A modifier that defines a protected initializer function that can be invoked at most once. In its scope, onlyInitializing functions can be used to initialize parent contracts.

Similar to reinitializer(1), except that functions marked with initializer can be nested in the context of a constructor.

Emits an [Initialized](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Initializable-Initialized-uint8-) event.

**reinitializer(uint8 version)**modifier

A modifier that defines a protected reinitializer function that can be invoked at most once, and only if the contract hasn’t been initialized to a greater version before. In its scope, onlyInitializing functions can be used to initialize parent contracts.

A reinitializer may be used after the original initialization step. This is essential to configure modules that are added through upgrades and that require initialization.

When version is 1, this modifier is similar to initializer, except that functions marked with reinitializer cannot be nested. If one is invoked in the context of another, execution will revert.

Note that versions can jump in increments greater than 1; this implies that if multiple reinitializers coexist in a contract, executing them in the right order is up to the developer or operator.

|  |  |
| --- | --- |
|  | setting the version to 255 will prevent any future reinitialization. |

Emits an [Initialized](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Initializable-Initialized-uint8-) event.

**onlyInitializing()**modifier

Modifier to protect an initialization function so that it can only be invoked by functions with the [initializer](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Initializable-initializer--) and [reinitializer](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Initializable-reinitializer-uint8-) modifiers, directly or indirectly.

**\_disableInitializers()**internal

Locks the contract, preventing any future reinitialization. This cannot be part of an initializer call. Calling this in the constructor of a contract will prevent that contract from being initialized or reinitialized to any version. It is recommended to use this to lock implementation contracts that are designed to be called through proxies.

Emits an [Initialized](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Initializable-Initialized-uint8-) event the first time it is successfully executed.

**\_getInitializedVersion() → uint8**internal

Returns the highest version that has been initialized. See [reinitializer](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Initializable-reinitializer-uint8-).

**\_isInitializing() → bool**internal

Returns true if the contract is currently initializing. See [onlyInitializing](https://docs.openzeppelin.com/contracts/4.x/api/proxy#Initializable-onlyInitializing--).

**Initialized(uint8 version)**event

Triggered when the contract has been initialized or reinitialized.

**UUPSUpgradeable**

import "@openzeppelin/contracts/proxy/utils/UUPSUpgradeable.sol";

An upgradeability mechanism designed for UUPS proxies. The functions included here can perform an upgrade of an [ERC1967Proxy](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Proxy), when this contract is set as the implementation behind such a proxy.

A security mechanism ensures that an upgrade does not turn off upgradeability accidentally, although this risk is reinstated if the upgrade retains upgradeability but removes the security mechanism, e.g. by replacing UUPSUpgradeable with a custom implementation of upgrades.

The [\_authorizeUpgrade](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UUPSUpgradeable-_authorizeUpgrade-address-) function must be overridden to include access restriction to the upgrade mechanism.

*Available since v4.1.*

**MODIFIERS**

* [onlyProxy()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UUPSUpgradeable-onlyProxy--)
* [notDelegated()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UUPSUpgradeable-notDelegated--)

**FUNCTIONS**

* [proxiableUUID()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UUPSUpgradeable-proxiableUUID--)
* [upgradeTo(newImplementation)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UUPSUpgradeable-upgradeTo-address-)
* [upgradeToAndCall(newImplementation, data)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UUPSUpgradeable-upgradeToAndCall-address-bytes-)
* [\_authorizeUpgrade(newImplementation)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UUPSUpgradeable-_authorizeUpgrade-address-)

**ERC1967UPGRADE**

* [\_getImplementation()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_getImplementation--)
* [\_upgradeTo(newImplementation)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeTo-address-)
* [\_upgradeToAndCall(newImplementation, data, forceCall)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeToAndCall-address-bytes-bool-)
* [\_upgradeToAndCallUUPS(newImplementation, data, forceCall)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeToAndCallUUPS-address-bytes-bool-)
* [\_getAdmin()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_getAdmin--)
* [\_changeAdmin(newAdmin)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_changeAdmin-address-)
* [\_getBeacon()](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_getBeacon--)
* [\_upgradeBeaconToAndCall(newBeacon, data, forceCall)](https://docs.openzeppelin.com/contracts/4.x/api/proxy#ERC1967Upgrade-_upgradeBeaconToAndCall-address-bytes-bool-)

**EVENTS**

**IERC1967**

* [Upgraded(implementation)](https://docs.openzeppelin.com/contracts/4.x/api/interfaces#IERC1967-Upgraded-address-)
* [AdminChanged(previousAdmin, newAdmin)](https://docs.openzeppelin.com/contracts/4.x/api/interfaces#IERC1967-AdminChanged-address-address-)
* [BeaconUpgraded(beacon)](https://docs.openzeppelin.com/contracts/4.x/api/interfaces#IERC1967-BeaconUpgraded-address-)

**onlyProxy()**modifier

Check that the execution is being performed through a delegatecall call and that the execution context is a proxy contract with an implementation (as defined in ERC1967) pointing to self. This should only be the case for UUPS and transparent proxies that are using the current contract as their implementation. Execution of a function through ERC1167 minimal proxies (clones) would not normally pass this test, but is not guaranteed to fail.

**notDelegated()**modifier

Check that the execution is not being performed through a delegate call. This allows a function to be callable on the implementing contract but not through proxies.

**proxiableUUID() → bytes32**external

Implementation of the ERC1822 [proxiableUUID](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UUPSUpgradeable-proxiableUUID--) function. This returns the storage slot used by the implementation. It is used to validate the implementation’s compatibility when performing an upgrade.

|  |  |
| --- | --- |
|  | A proxy pointing at a proxiable contract should not be considered proxiable itself, because this risks bricking a proxy that upgrades to it, by delegating to itself until out of gas. Thus it is critical that this function revert if invoked through a proxy. This is guaranteed by the notDelegated modifier. |

**upgradeTo(address newImplementation)**external

Upgrade the implementation of the proxy to newImplementation.

Calls [\_authorizeUpgrade](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UUPSUpgradeable-_authorizeUpgrade-address-).

Emits an [Upgraded](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UpgradeableBeacon-Upgraded-address-) event.

**upgradeToAndCall(address newImplementation, bytes data)**external

Upgrade the implementation of the proxy to newImplementation, and subsequently execute the function call encoded in data.

Calls [\_authorizeUpgrade](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UUPSUpgradeable-_authorizeUpgrade-address-).

Emits an [Upgraded](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UpgradeableBeacon-Upgraded-address-) event.

**\_authorizeUpgrade(address newImplementation)**internal

Function that should revert when msg.sender is not authorized to upgrade the contract. Called by [upgradeTo](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UUPSUpgradeable-upgradeTo-address-) and [upgradeToAndCall](https://docs.openzeppelin.com/contracts/4.x/api/proxy#UUPSUpgradeable-upgradeToAndCall-address-bytes-).

Normally, this function will use an [access control](https://docs.openzeppelin.com/contracts/4.x/api/access) modifier such as [Ownable.onlyOwner](https://docs.openzeppelin.com/contracts/4.x/api/access#Ownable-onlyOwner--).

function \_authorizeUpgrade(address) internal override onlyOwner {}