

```

1 // SPDX-License-Identifier: MIT
2 // OpenZeppelin Contracts v4.4.1 (proxy/transparent/ProxyAdmin.sol)
3
4 pragma solidity ^0.8.0;
5
6 import "../TransparentUpgradeableProxy.sol";
7 import "../../access/Ownable.sol";
8
9 /**
10  * @dev This is an auxiliary contract meant to be assigned as the admin of a
11  * {TransparentUpgradeableProxy}. For an
12  * explanation of why you would want to use this see the documentation for
13  * {TransparentUpgradeableProxy}.
14  */
15 contract ProxyAdmin is Ownable {
16     /**
17      * @dev Returns the current implementation of `proxy`.
18      *
19      * Requirements:
20      * - This contract must be the admin of `proxy`.
21      */
22     function getProxyImplementation(TransparentUpgradeableProxy proxy) public view
23     virtual returns (address) {
24         // We need to manually run the static call since the getter cannot be flagged
25         // as view
26         // bytes4(keccak256("implementation()")) == 0x5c60da1b
27         (bool success, bytes memory returndata) = address(proxy).staticcall(hex
28         "5c60da1b");
29         require(success);
30         return abi.decode(returndata, (address));
31     }
32
33     /**
34      * @dev Returns the current admin of `proxy`.
35      *
36      * Requirements:
37      * - This contract must be the admin of `proxy`.
38      */
39     function getProxyAdmin(TransparentUpgradeableProxy proxy) public view virtual
40     returns (address) {
41         // We need to manually run the static call since the getter cannot be flagged
42         // as view
43         // bytes4(keccak256("admin()")) == 0xf851a440
44         (bool success, bytes memory returndata) = address(proxy).staticcall(hex
45         "f851a440");
46         require(success);
47         return abi.decode(returndata, (address));
48     }
49
50     /**
51      * @dev Changes the admin of `proxy` to `newAdmin`.
52      *
53      * Requirements:
54      * - This contract must be the current admin of `proxy`.
55      */
56     function changeProxyAdmin(TransparentUpgradeableProxy proxy, address newAdmin)
57     public virtual onlyOwner {
58         proxy.changeAdmin(newAdmin);
59     }
60
61     /**
62      * @dev Upgrades `proxy` to `implementation`. See
63      * {TransparentUpgradeableProxy-upgradeTo}.
64      *
65      * Requirements:
66      * - This contract must be the admin of `proxy`.
67      */
68     function upgrade(TransparentUpgradeableProxy proxy, address implementation) public
69     virtual onlyOwner {

```

```
63         proxy.upgradeTo(implementation);
64     }
65
66     /**
67     * @dev Upgrades `proxy` to `implementation` and calls a function on the new
68     * implementation. See
69     * {TransparentUpgradeableProxy-upgradeToAndCall}.
70     *
71     * Requirements:
72     * - This contract must be the admin of `proxy`.
73     */
74     function upgradeAndCall(
75         TransparentUpgradeableProxy proxy,
76         address implementation,
77         bytes memory data
78     ) public payable virtual onlyOwner {
79         proxy.upgradeToAndCall{value: msg.value}(implementation, data);
80     }
81 }
82
```