合约升级

登链社区 - Tiny熊

合约有问题怎么办?

- 合约一旦部署，便不可更改
- 合约中有错误如何修复?
- 要添加额外功能，要怎么办?

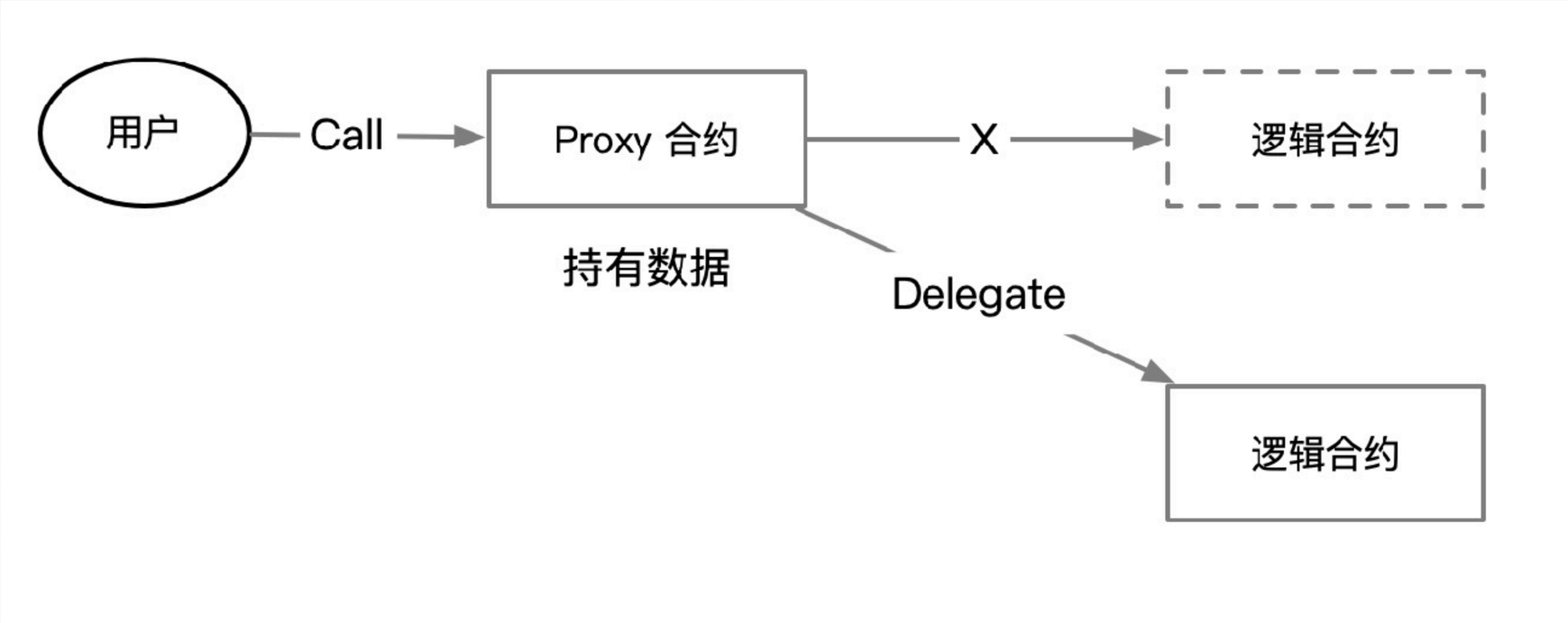
问题?

- 没有升级的情况下，如何做迁移?

合约迁移

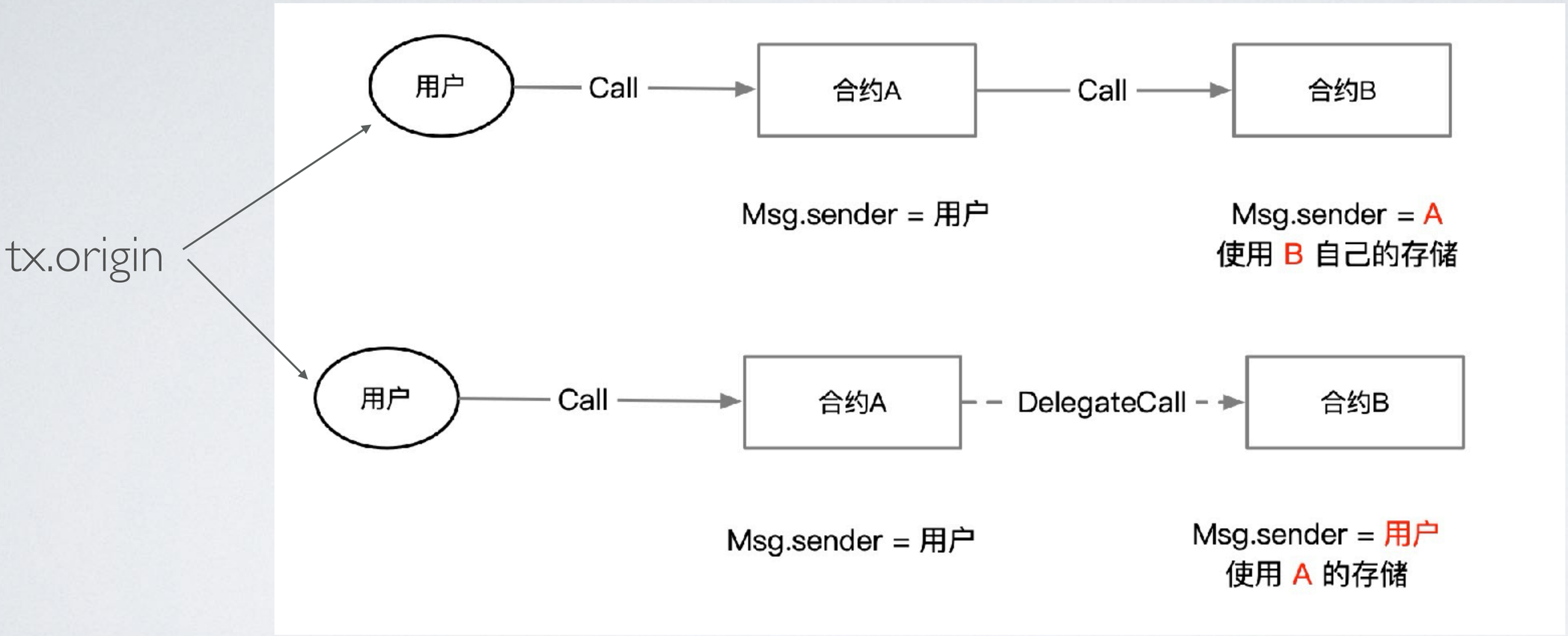
- 先禁用老合约（如果可以）
- 编写新合约、读取老合约的状态（如果可以）
- 确定 Bug 所在区块高度，链下索引所有数据
  - 一次性迁移：如发送等量的新币给所有用户 （适合数据量小）
  - 逐步迁移：用户自行迁移，防止重复执行
- 合约更新后，修改前后端、通知用户、生态伙伴

合约升级



address impl

Call 与 delegateCall 回顾



testCall_Delegate.sol

尝试给 Counter 升级

```
pragma solidity ^0.8.0;

contract Counter {
    uint private counter;

    function add(uint256 i) public {
        counter += 1;
    }

    function get() public view returns(uint) {
        return counter;
    }
}
```

升级尝试 1 – 使用代理

```solidity
pragma solidity ^0.8.0;

contract CounterProxy {
    address impl;
    uint private counter;

    function add(uint256 i) public {
        bytes memory callData = abi.encodeWithSignature("add(uint256)", n);
        (bool ok,) = address(impl).delegatecall(callData);
        if(!ok) revert("Delegate call failed");
    }

    function get() public view returns(uint) {
        bytes memory callData = abi.encodeWithSignature("get()");
        (bool ok, bytes memory retVal) = address(impl).delegatecall(callData);
        if(!ok) revert("Delegate call failed");
        return abi.decode(retVal, (uint256));
    }
}
```

CounterProxy.sol

升级尝试 1 – 使用代理

- 问题 1：代理和逻辑合约的存储布局不一致发生无法预期的错误(存储布局对齐)
- 问题 2：升级后，想添加新方法，怎么办?

CounterProxy.sol

CounterProxy

Storage	
slot	变量
0	impl
1	counter

Counter.sol

Storage	
slot	变量
0	counter

升级 1

- 需要解决存储布局冲突
  - 实现合约地址槽位：bytes32(uint(keccak256("eip1967.proxy.implementation")) - 1)
  - 升级添加的变量，必须在末尾添加

- 如何委托未来添加的函数及获取返回值?

升级尝试 2 – 统一委托

- fallback 统一委托
- 用汇编魔法获取返回值

CounterFallback.sol

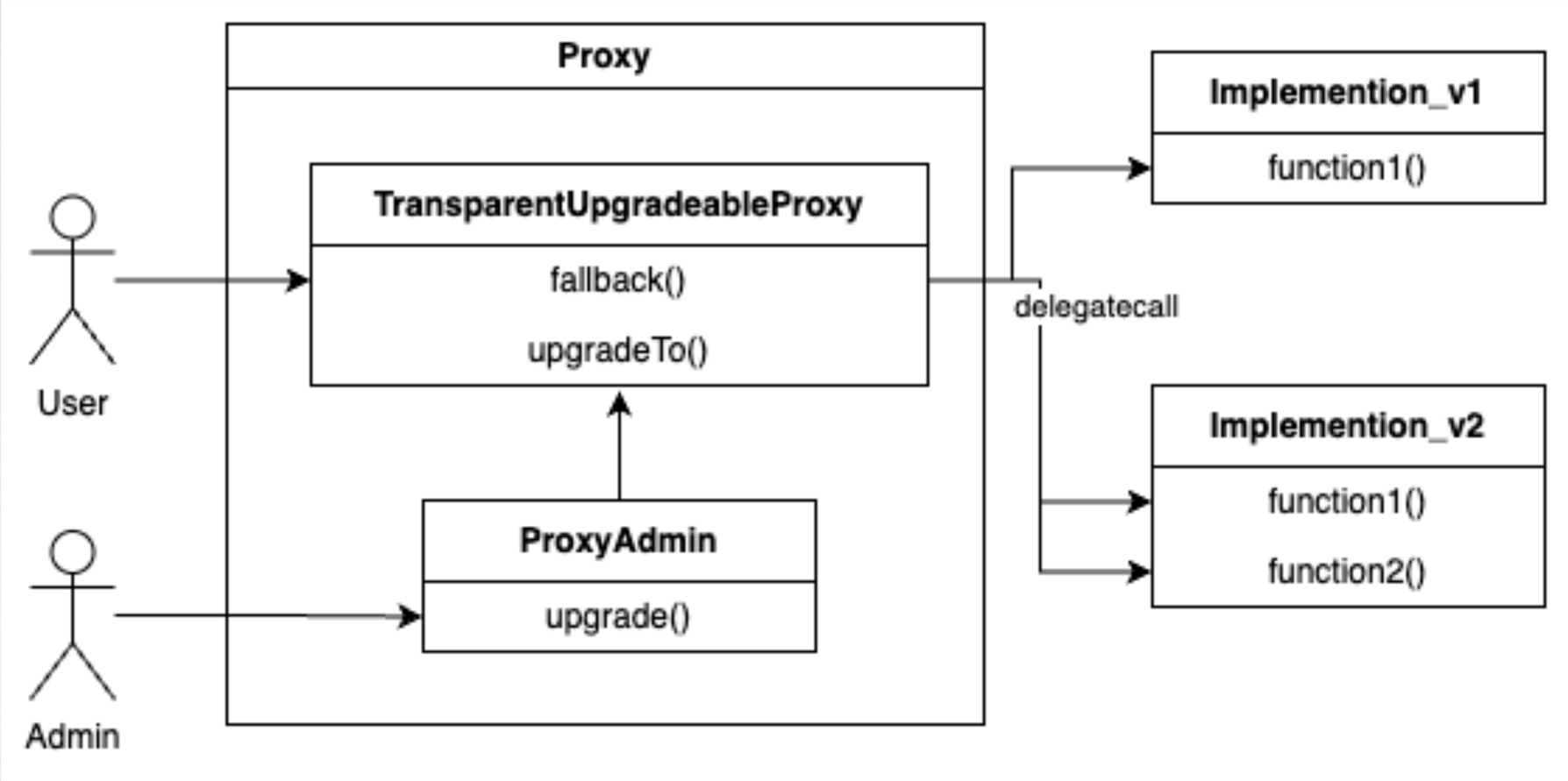
```
        assembly {
            //获得自由空闲指针
            let ptr := mload(0x40)
            //将返回值从返回缓冲去copy到指针所指位置
            returndatacopy(ptr, 0, returndatasize())

            //根据是否调用成功决定是返回数据还是直接revert整个函数
            switch success
            case 0 { revert(ptr, returndatasize()) }
            default { return(ptr, returndatasize()) }
        }
```

升级尝试 – 通用升级

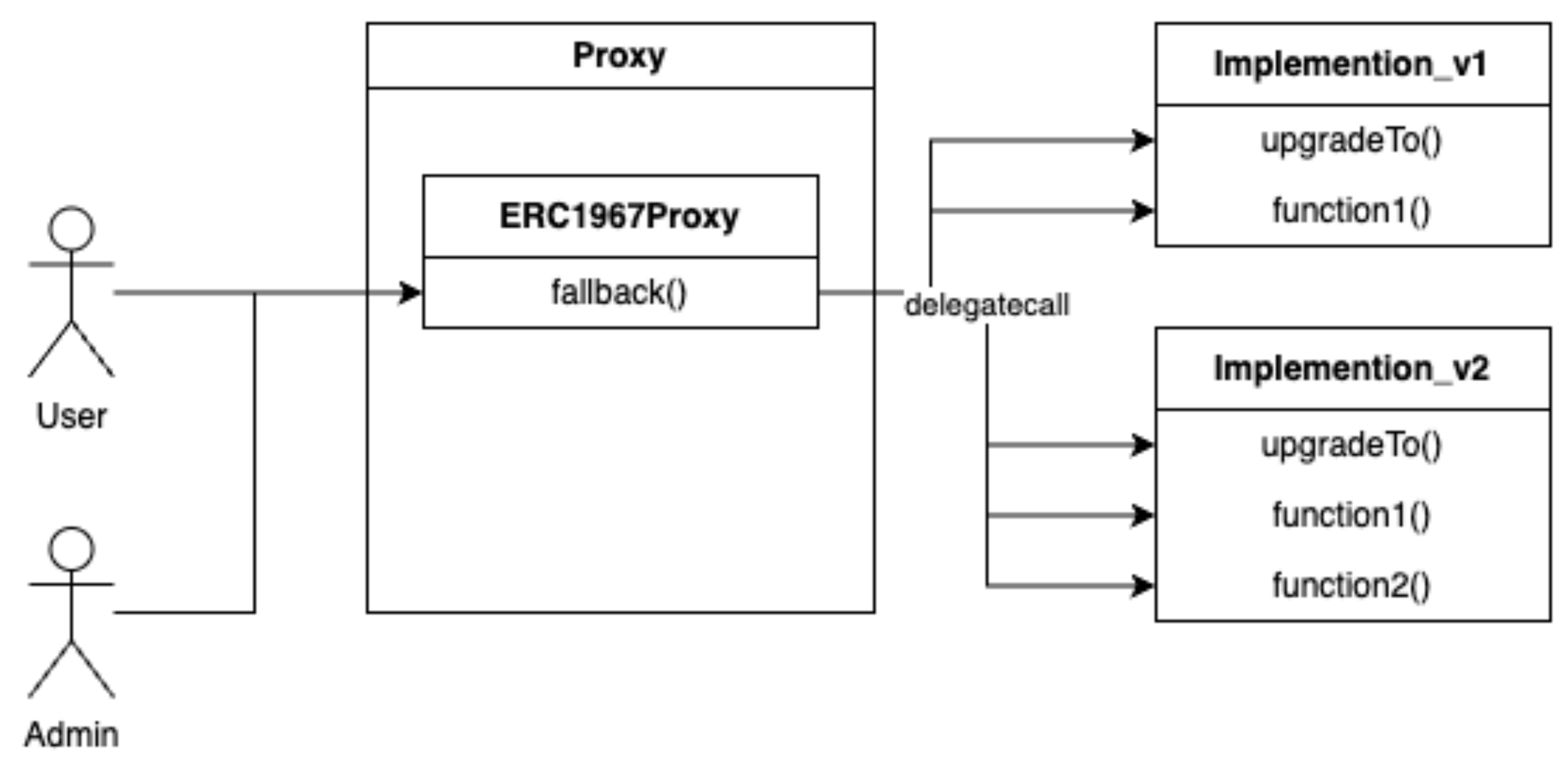
- 函数冲撞问题，若某个函数与upgradeTo() 函数选择器一样会出现什么问题?
- 透明代理（Transparent Proxy）- ERC1967Proxy
- UUPS（universal upgradeable proxy standard）- ERC-1822
- 钻石代理（Diamonds, Multi-Facet Proxy） - ERC-2535
- 信标代理（Beacon Proxy）

透明代理



TransparentProxy.sol

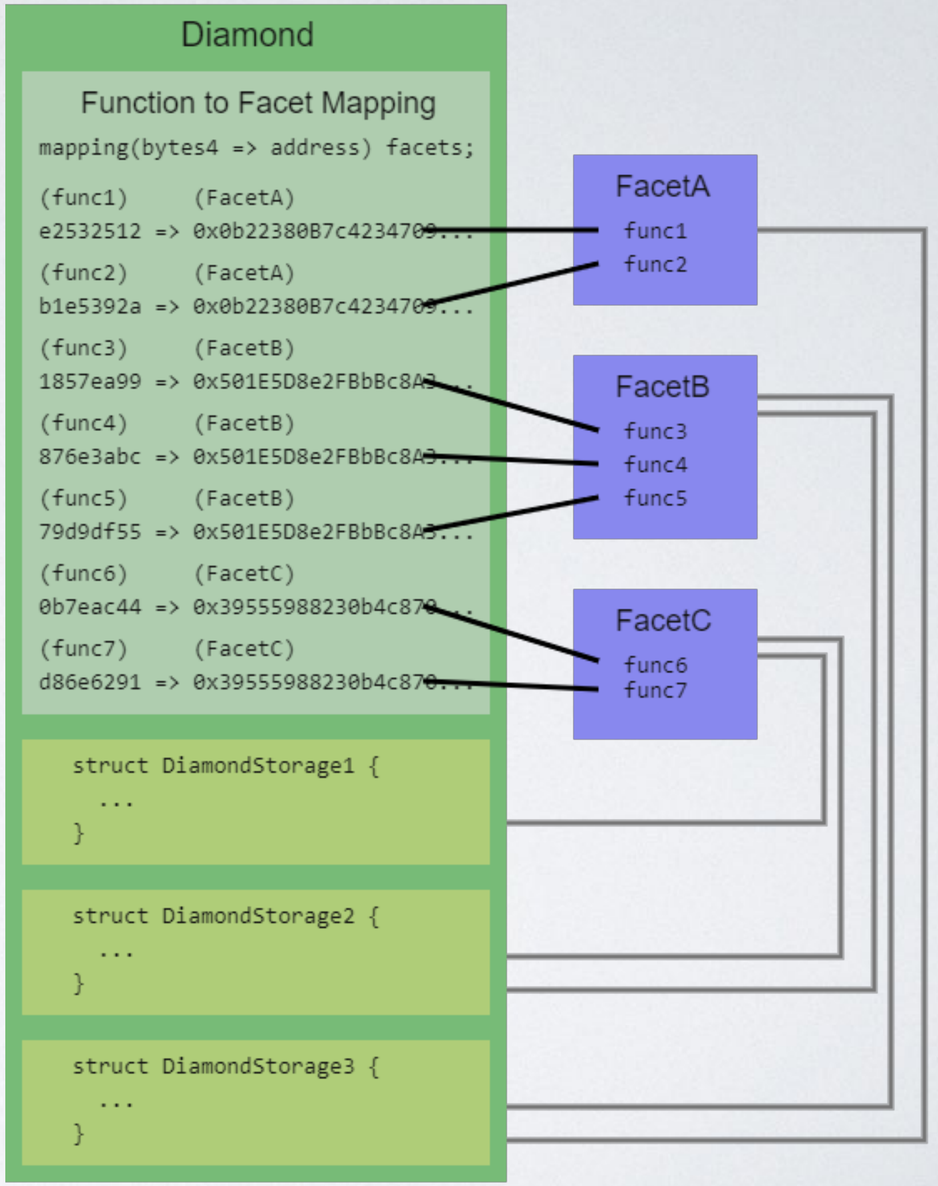
UUPS



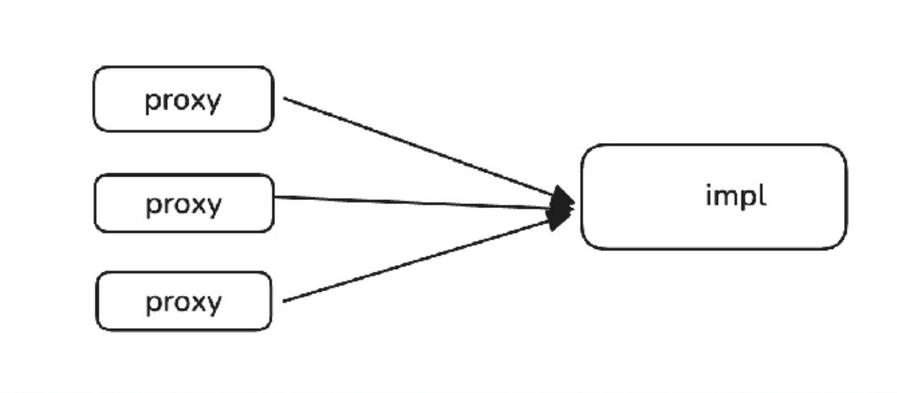
UUPSProxy.sol

钻石代理

适合大型合约的情况
代理像一个钻石，反射到不同的实现

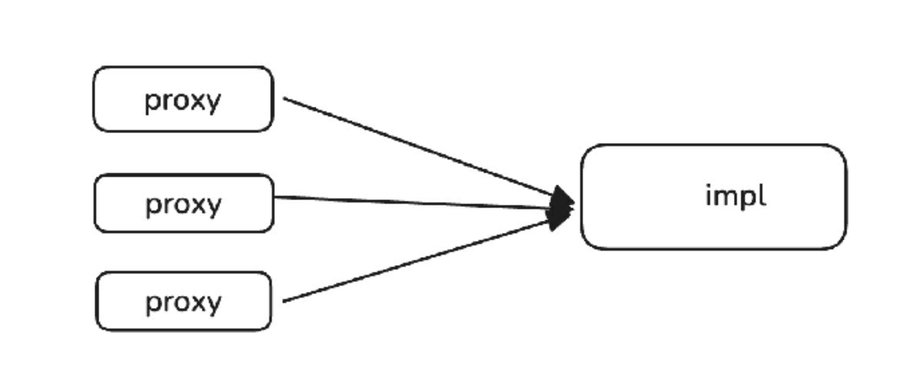


最小代理部署的合约如何升级?

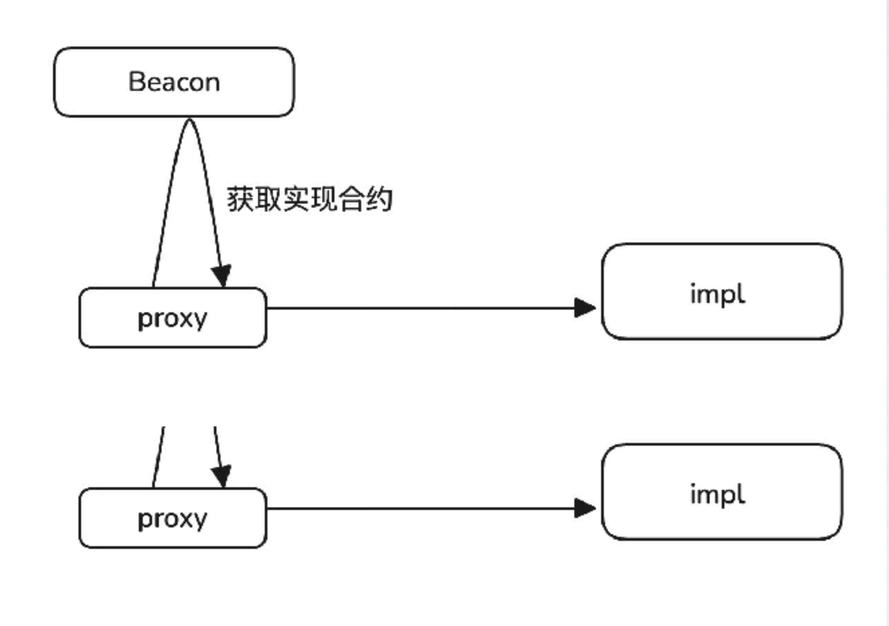


如何替换 impl ?

信标代理



最小代理工厂



适合大量部署的情况

合约升级


合约升级
重温 Call/DelegateCall
解决1. 代理和逻辑合约存储布局问题
解决2： 函数调用及返回值泛化
解决 3： 函数冲撞

使用 DelegateCall 要注意的点

- 代理和逻辑合约的存储布局需要一致。
- delegateCall 返回值
  - (bool success, bytes memory returnData) = address.delegatecall(payload);
  - Bytes 需转化为具体的类型
- 不能有函数冲撞
- 初始化问题？ - 实现合约中构造函数无效

实际开发中使用升级

- 通常仅需要关注实现合约
  - 注意构造函数替换，如修改为 Initializable，避免使用合约级初始化
  - 复用可升级合约：https://docs.openzeppelin.com/contracts/5.x/upgradeable
- 使用 contracts-upgradeable Hardhat/Foundry 插件， 自动完成代理合约部署

开发中使用升级（Hardhat）

- hardhat-upgrades
  - npm install --save-dev @openzeppelin/hardhat-upgrades
- contracts-upgradeable
  - npm install —save-dev @openzeppelin/contracts-upgradeable

https://github.com/xilibi2003/training_camp_2/tree/main/w3_2_code

开发中使用升级Foundry

- foundry-upgrades
  - forge install OpenZeppelin/openzeppelin-foundry-upgrades
- openzeppelin-contracts-upgradeable
  - forge install OpenZeppelin/openzeppelin-contracts-upgradeable

https://github.com/OpenZeppelin/openzeppelin-foundry-upgrades

https://github.com/OpenSpace100/blockchain-tasks/tree/upgrade/w3_permit

upgrade 分支

使用升级注意事项

- 用户操作的是代理，但无法阻止直接和逻辑合约交互。
  - 逻辑合约状态的任何更改不会影响到代理，但是逻辑合约销毁除外。
- 如果已经使用了最小代理工厂，实现合约将无法在使用升级功能，此时可考虑使用 Beacon 模式

练习题

- 编写一个可升级的 ERC721 合约
- 编写一个可升级的 NFT 市场合约
  - 第一版本普通合约
  - 第二版本，加入离线签名上架 NFT 功能方法（签名内容：tokenId， 价格），实现用户一次性 setApproveAll 给 NFT 市场合约，每次上架时使用签名上架。
  - 部署到测试网，并开源到区块链浏览器（在 Readme.md 中备注代理合约及两个实现的合约地址）

https://decert.me/quests/ddbdd3c4-a633-49d7-adf9-34a6292ce3a8

练习题

- 理解合约升级涉及的存储布局
  - 编写可升级合约时，如果第一个版本逻辑实现合约中有一个 mapping（ uint -> User） users ; User 是一个结构体类型，请问在第二个版本的逻辑实现合约，可否在User 结构体里添加一个变量？ 请说出你的理解。
  - 编写可升级合约时，如果第一个版本逻辑实现合约中有一个数组 User[] users ; User 是一个结构体类型，请问在第二个版本的逻辑实现合约，可否在User 结构体里添加一个变量？ 请说出你的理解。
  - 编写可升级合约时，如果逻辑实现合约有继承关系，什么情况下能父合约里添加变量？ 你认为的最佳实践是什么？

https://decert.me/quests/8ea21ac0-fc65-414a-8afd-9507c0fa2d90

练习题(不用)

- 在以太坊上用 ERC20 模拟铭文铸造，创建可升级的工厂合约：
  - 方法1： deployInscription(string symbol, uint totalSupply, uint perMint)
  - 方法 2： mintInscription(address tokenAddr)
  - 第 2 个版本加入铸造费用（price）

https://decert.me/quests/ac607bb0-53b5-421f-a9df-f3db4a1495f2