

Figure 1. Creating and importing the proposed DSCoT workspace in Remix IDE

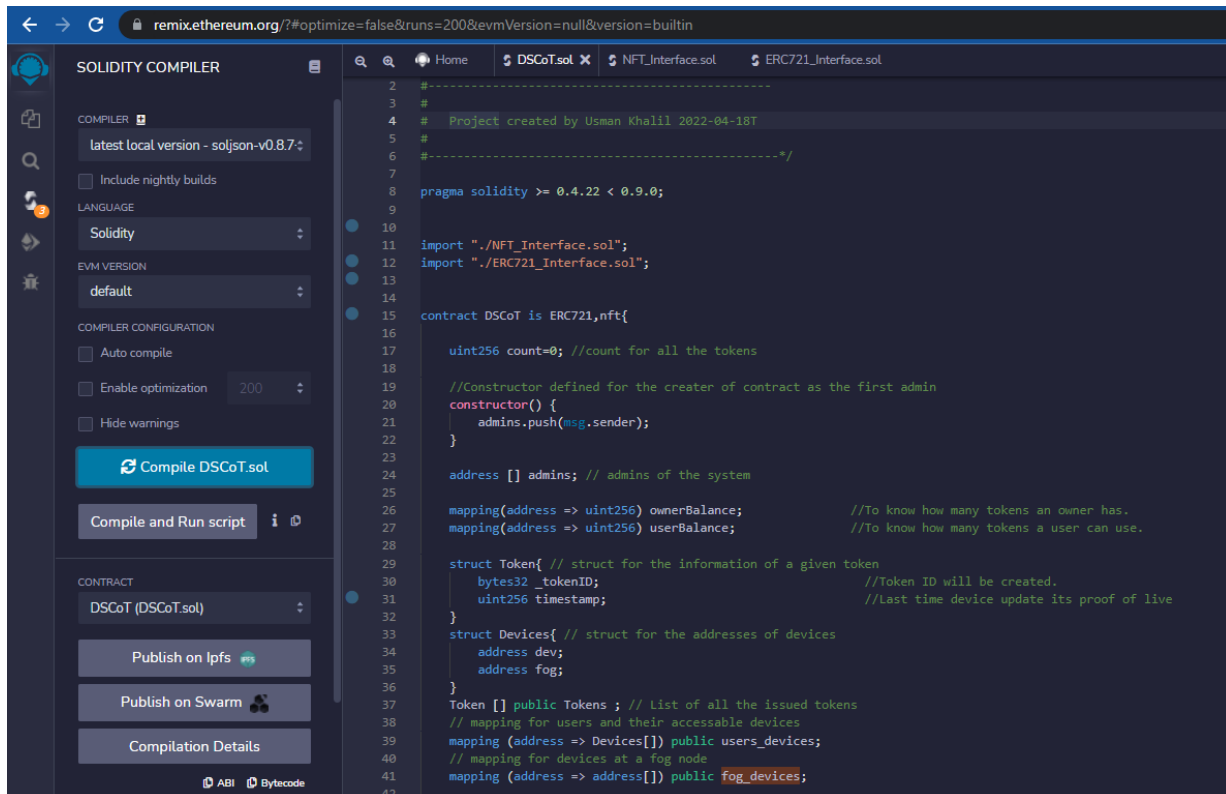


Figure 2. The proposed DSCoT Smart Contracts in Remix IDE

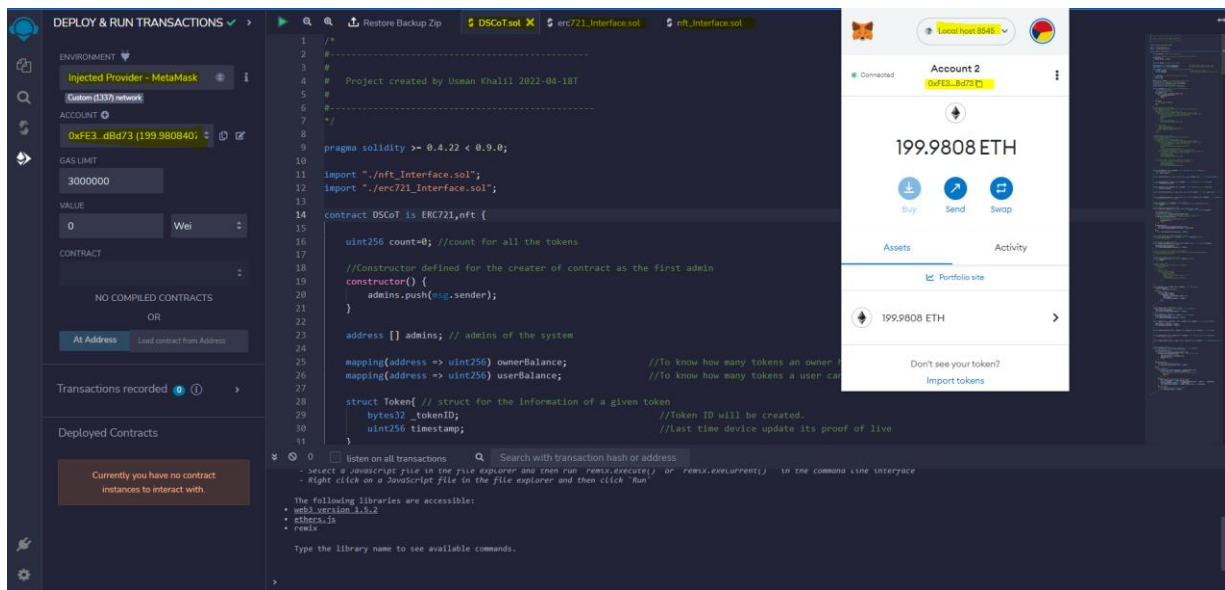


Figure 3. Digital Wallet ~ MetaMask with connected site ~ Remix IDE with the Proposed DSCoT SCs Compilation

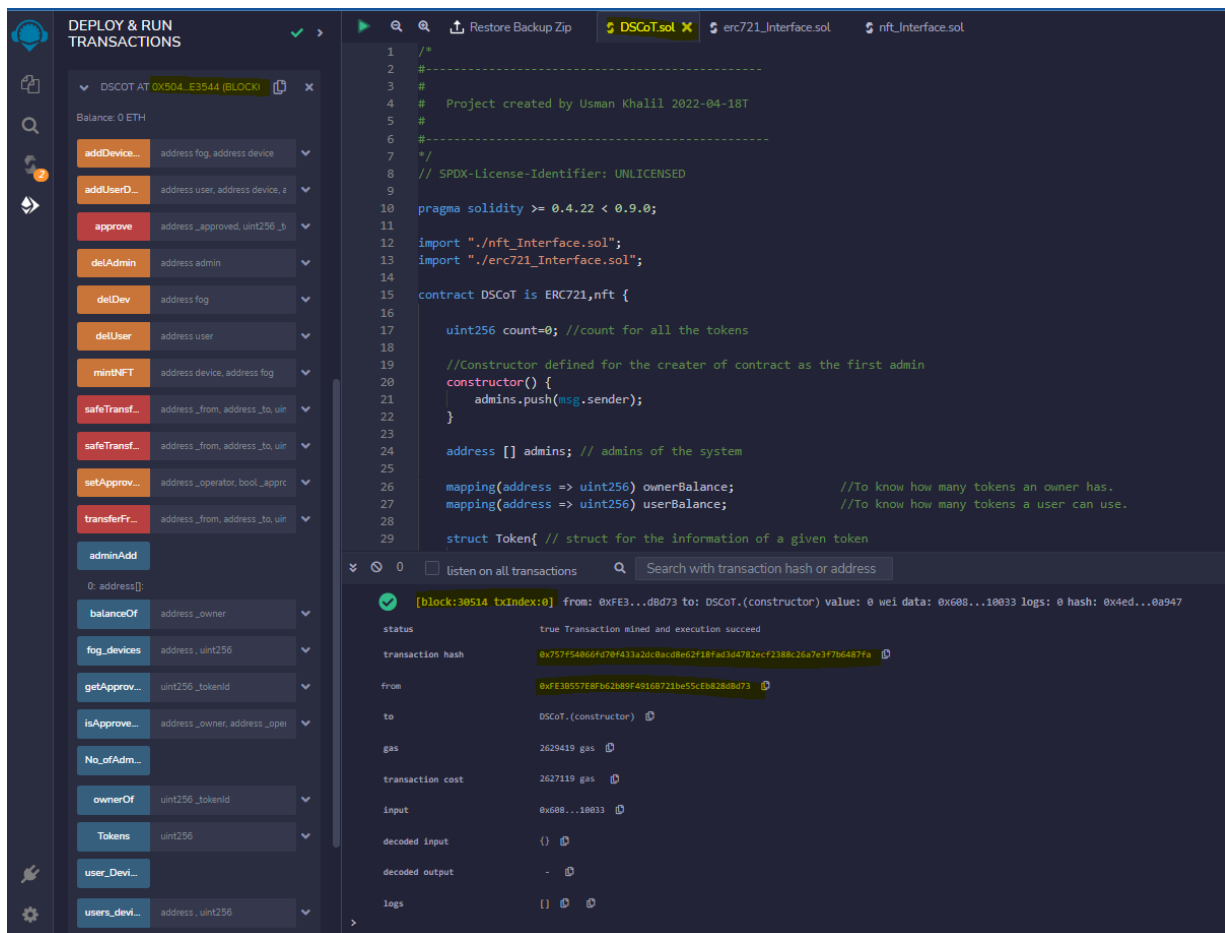


Figure 4. Deployment details of the proposed DSCoT SCs

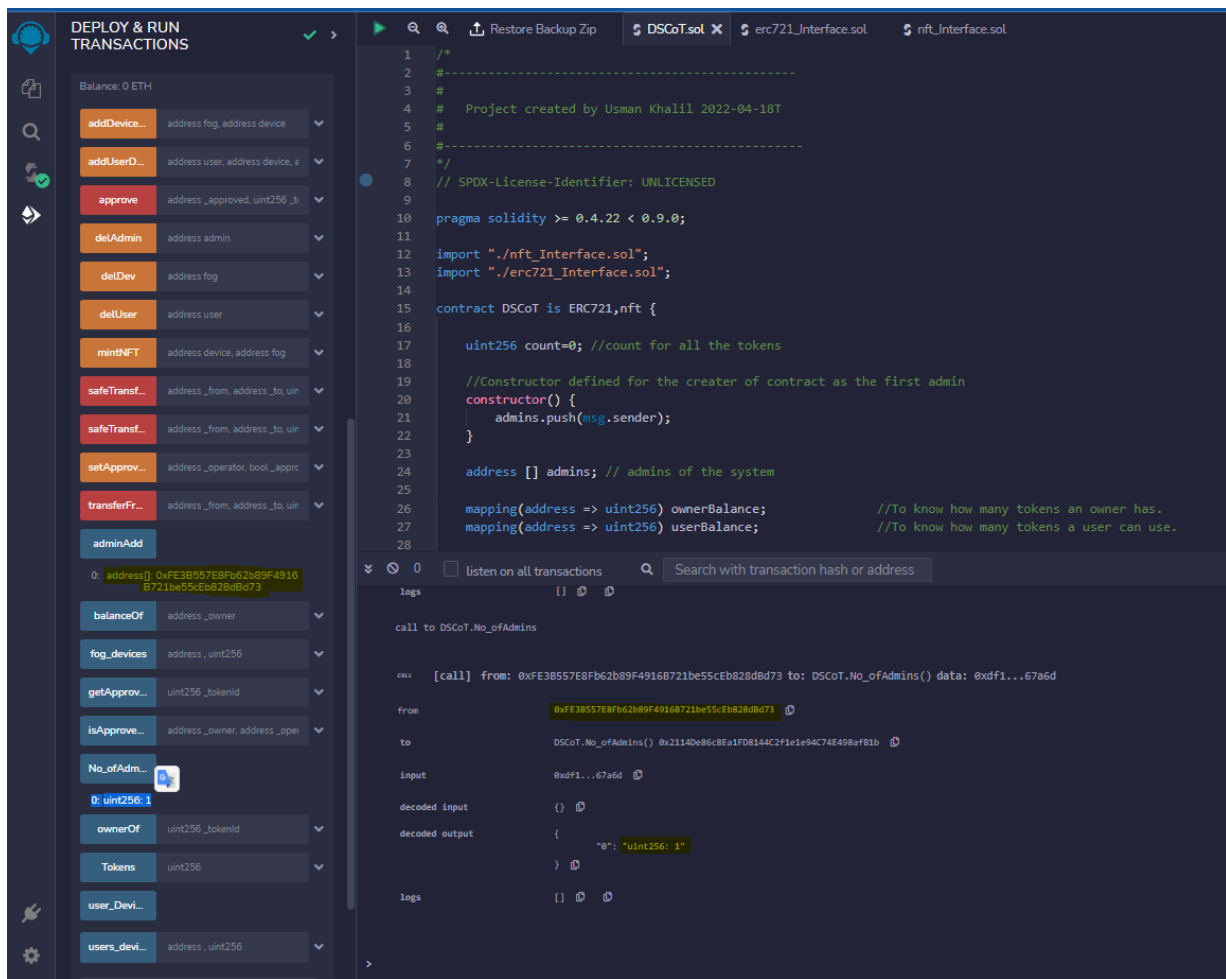


Figure 5. The proposed DSCoT ~ adminAdd() and No_ofAdmin () functions

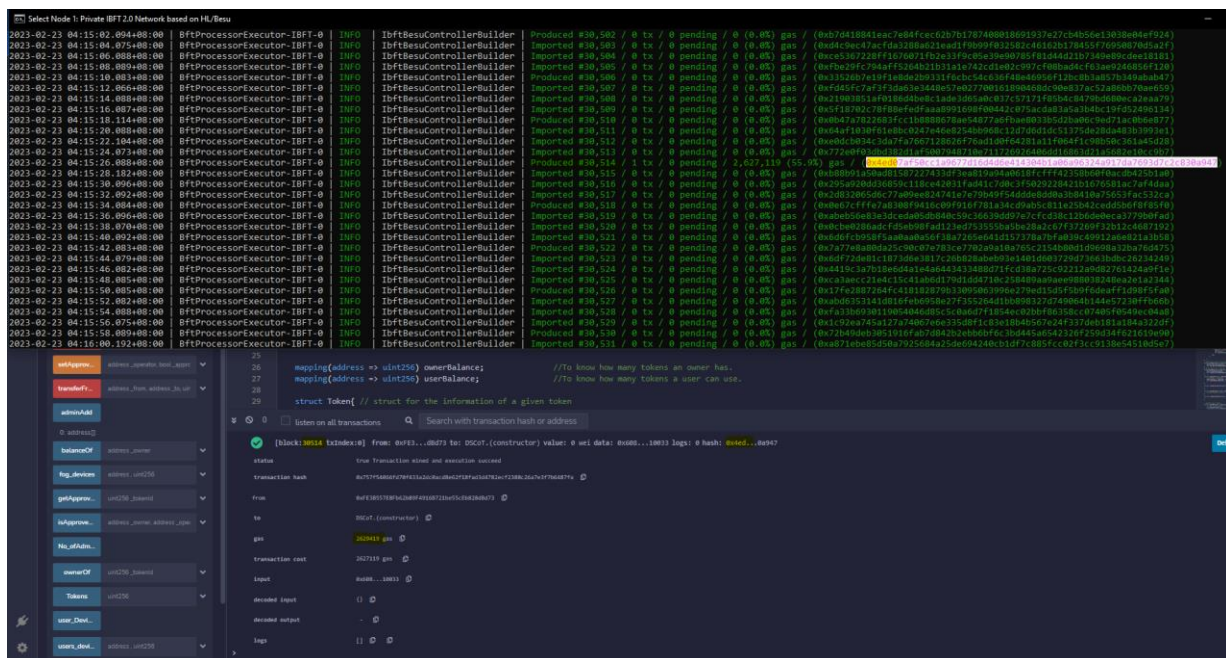


Figure 6. Tx confirmation on private Hyper Ledger Besu network

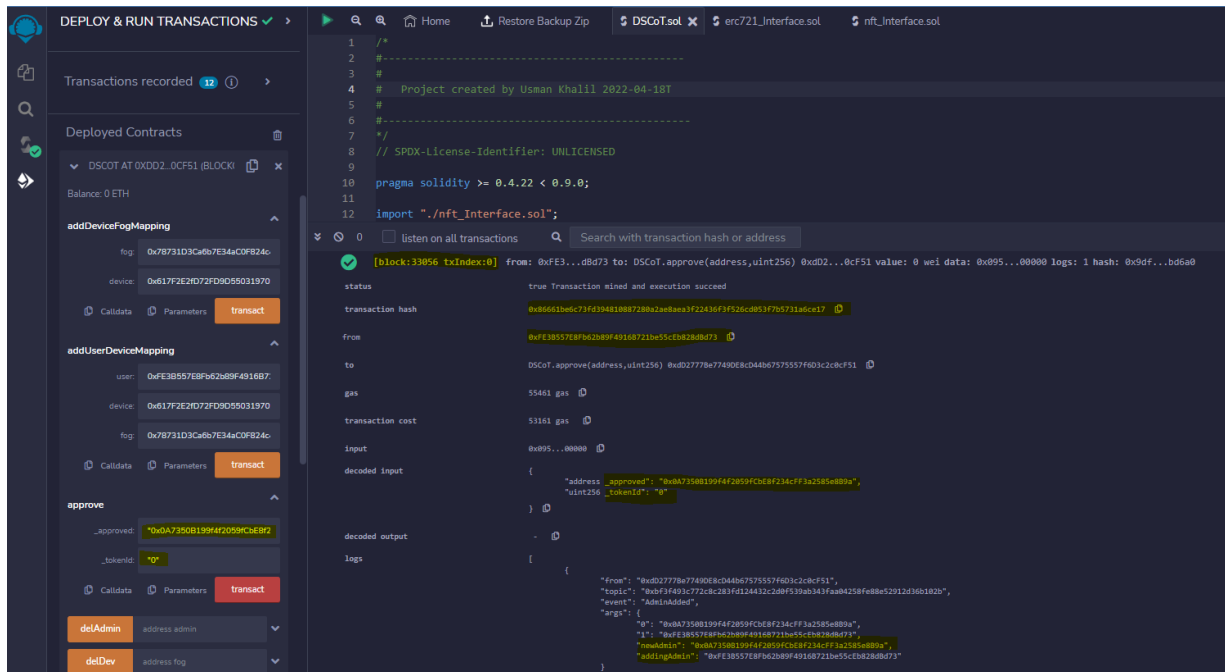


Figure 7. The proposed DSCoT ~ approve() function

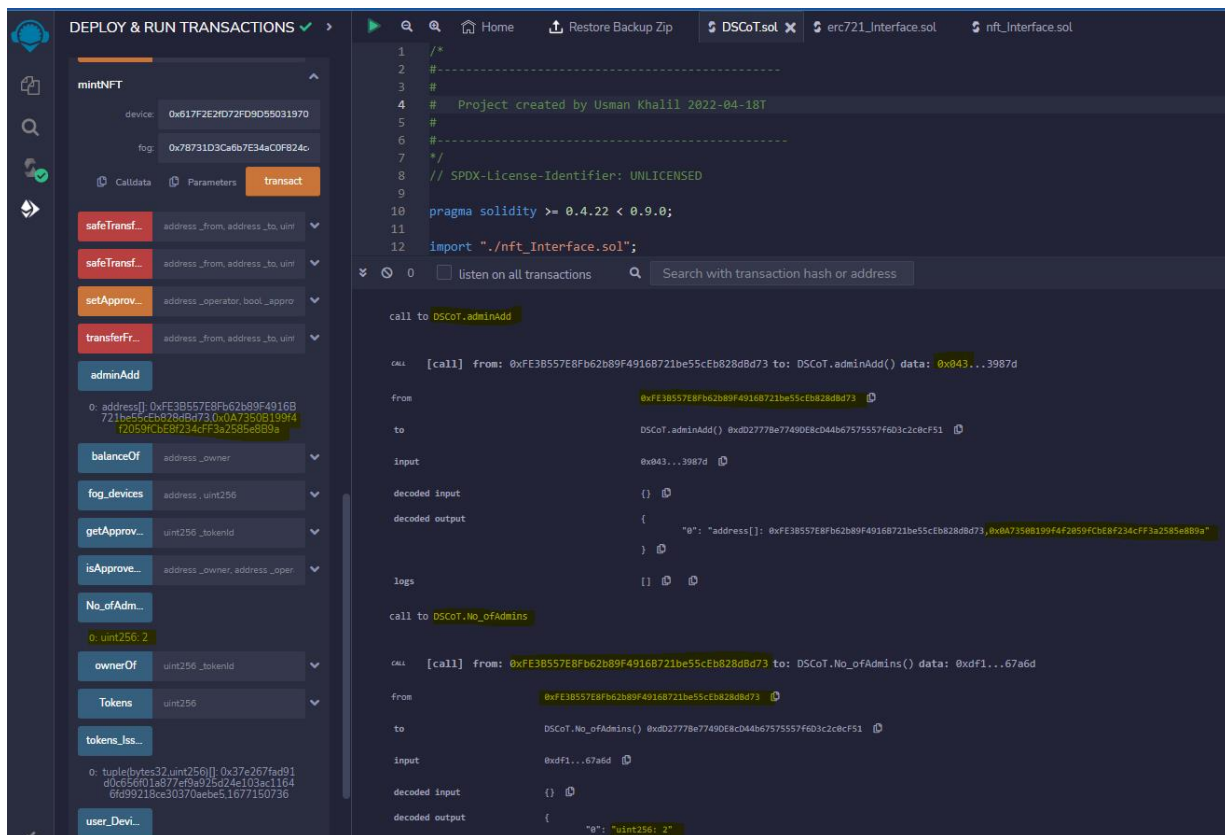


Figure 8. The proposed DSCoT ~ adminAdd() and No_ofAdmin () functions

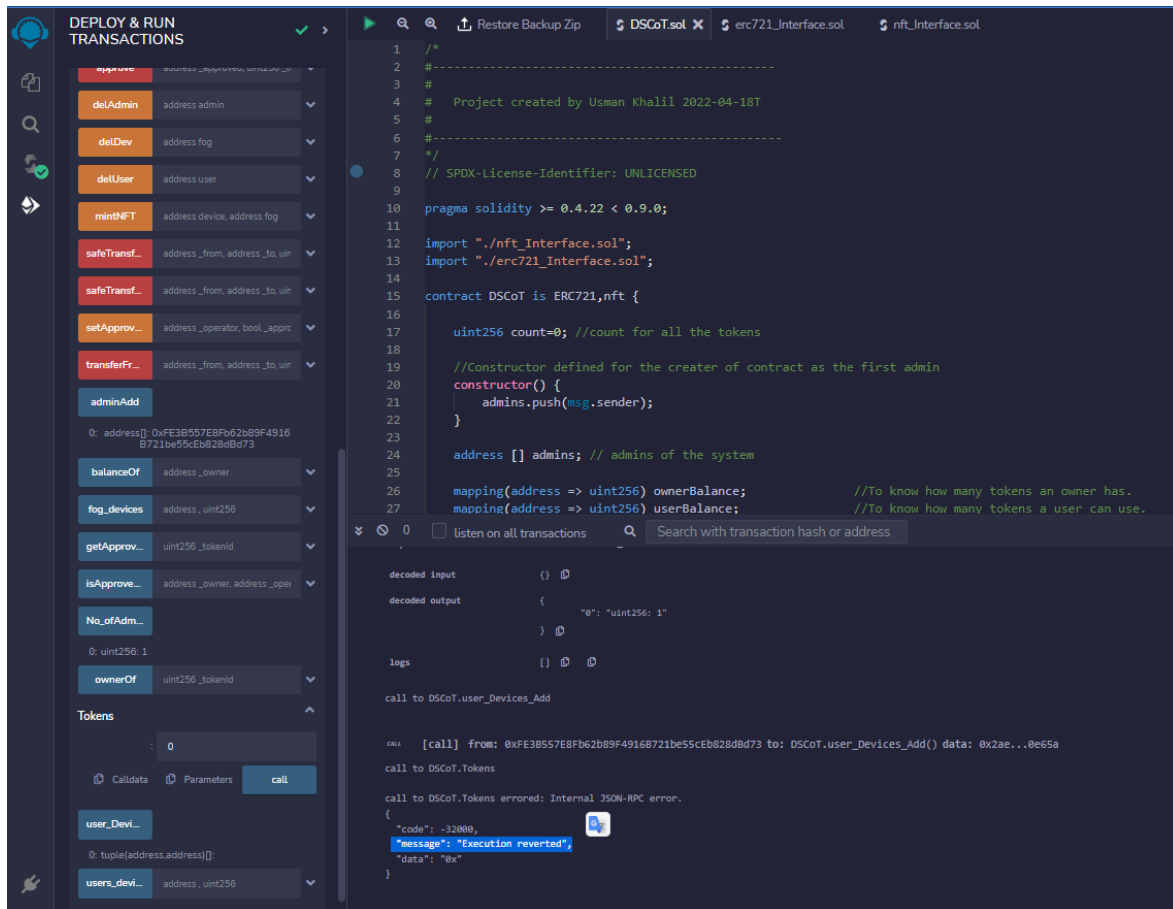


Figure 9. Tx denied as no payload exists

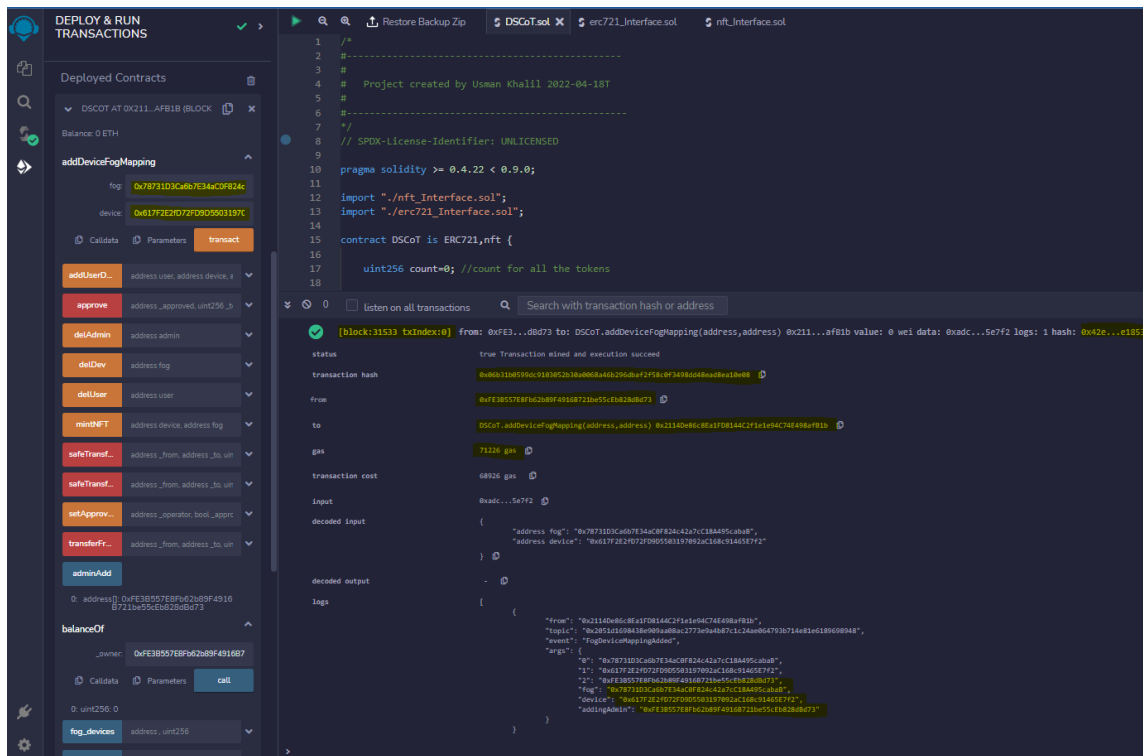


Figure 10. The proposed DSCoT ~ addDeviceFogMapping() function

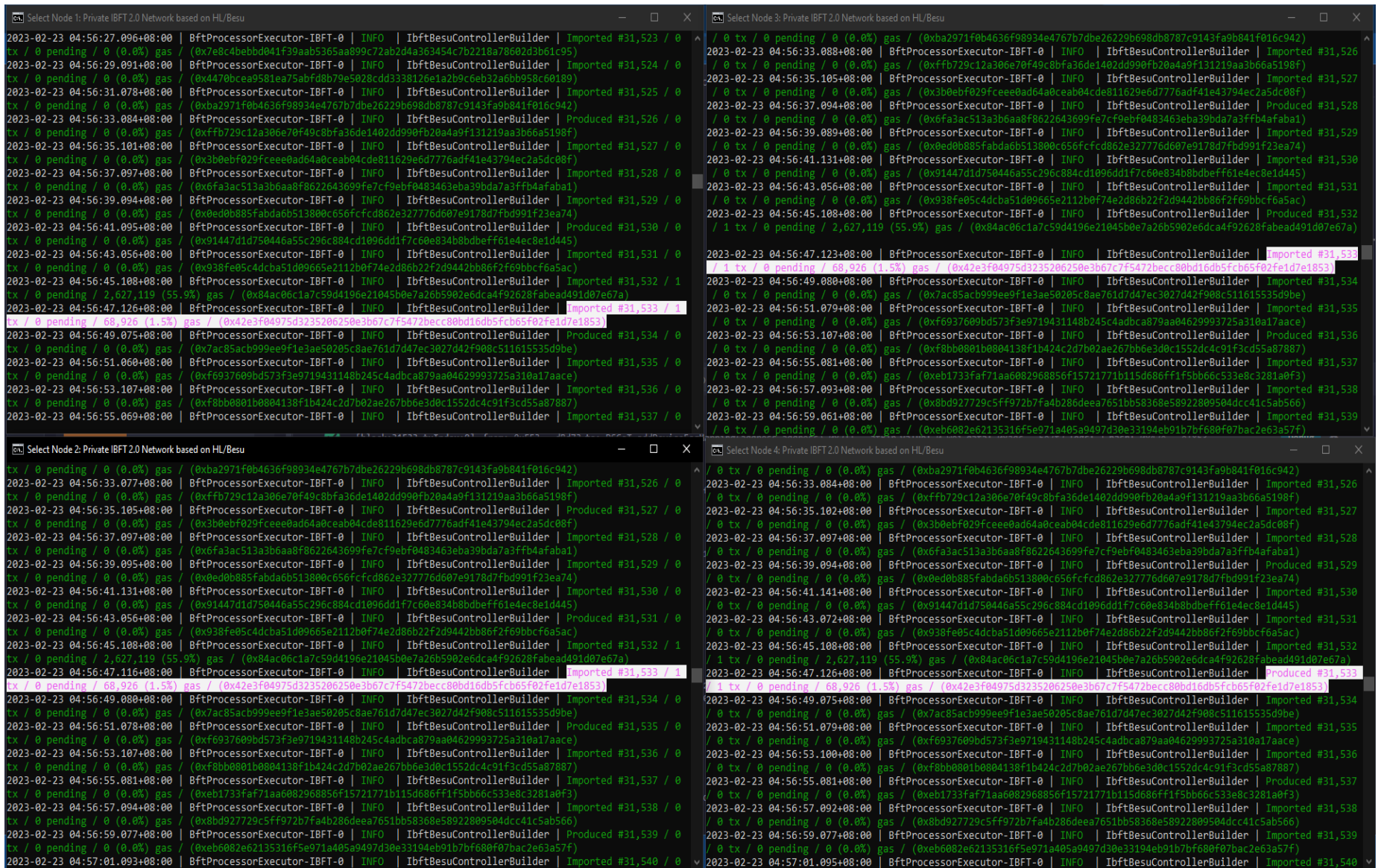


Figure 12. Tx addDeviceFogMapping() confirmation on all nodes in the private Hyper Ledger Besu network

The screenshot displays the Remix Ethereum IDE interface. The left sidebar shows the 'Deployed Contracts' section with a contract named 'DSCoT AT 0XD02...0CF51 (BLOCK)'. Below it, the 'addDeviceFogMapping' function is visible. The central code editor shows the Solidity code for the 'DSCoT' contract, including the 'addUserDeviceMapping' function. The right sidebar shows the 'Contract Interaction' details, including the status (Confirmed), transaction hash, and decoded input/output.

Figure 13. The proposed DSCoT ~ addUserDeviceMapping() function

The screenshot displays the Remix Ethereum IDE interface. The left sidebar shows the 'Deployed Contracts' section with a contract named 'DSCoT - contracts/DSCoT/DSCoT.sol'. Below it, the 'addDeviceFogMapping' function is visible. The central code editor shows the Solidity code for the 'DSCoT' contract, including the 'addUserDeviceMapping' function. The right sidebar shows the 'Contract Interaction' details, including the status (Confirmed), transaction hash, and decoded input/output.

Figure 14. The proposed DSCoT ~ addUserDeviceMapping() function

Figure 15. Tx `addUserDeviceMapping()` confirmation on all nodes in the private Hyper Ledger Besu network

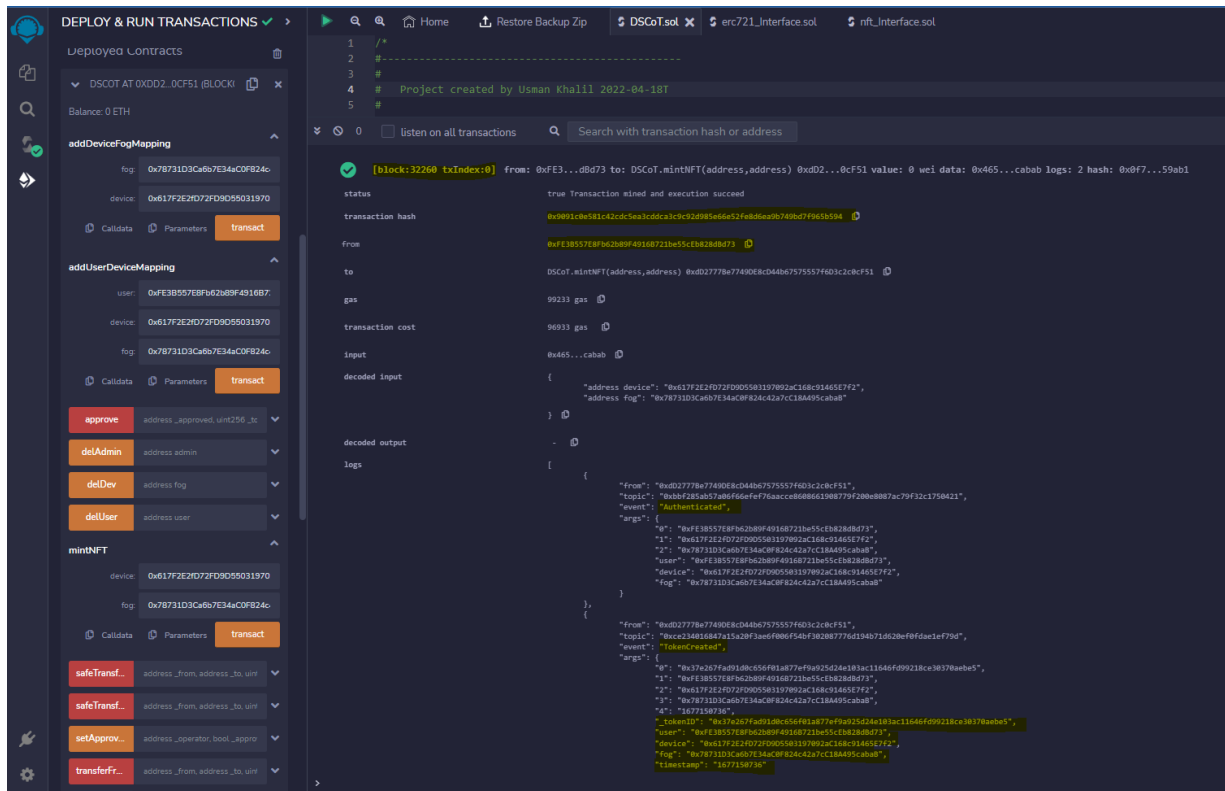


Figure 16. The proposed DSCoT ~ mintNFT() function

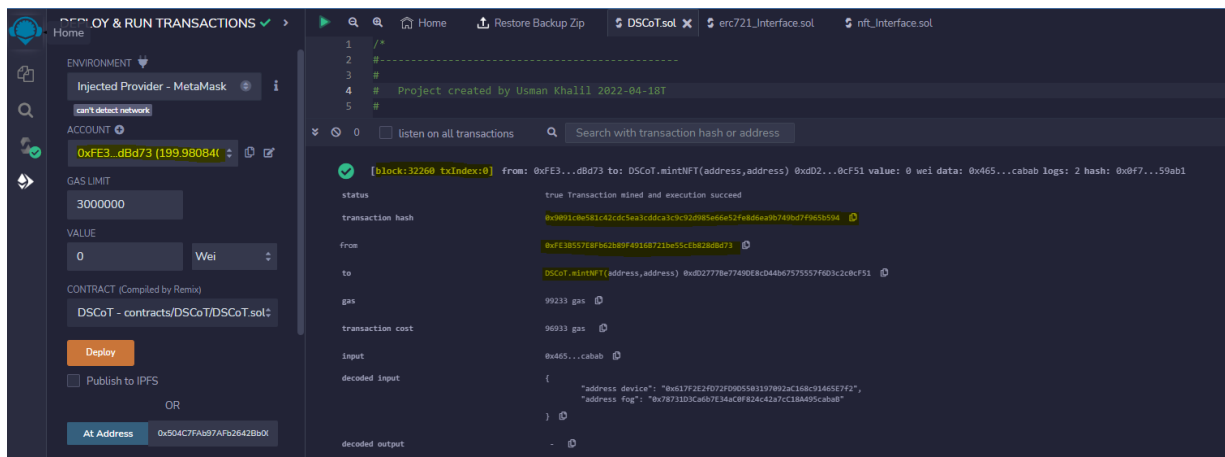


Figure 17. EOA_{owner} as EOA_{user} ~ mintNFT() function

The screenshot displays the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is active, showing a list of deployed contracts. The main panel displays the Solidity source code for the `DSCoT` contract, which is an ERC721 token. The code includes imports for `nft_Interface.sol` and `erc721_Interface.sol`, and defines the `DSCoT` contract with various functions and mappings.

```

7  //
8  // SPDX-License-Identifier: UNLICENSED
9
10 pragma solidity >= 0.4.22 < 0.9.0;
11
12 import "./nft_Interface.sol";
13 import "./erc721_Interface.sol";
14
15 contract DSCoT is ERC721, nft {
16
17     uint256 count=0; //count for all the tokens
18
19     //Constructor defined for the creator of contract as the first admin
20     constructor() {
21         admins.push(msg.sender);
22     }
23
24     address [] admins; // admins of the system
25
26     mapping(address => uint256) ownerBalance; //To know how many tokens an owner has.
27     mapping(address => uint256) userBalance; //To know how many tokens a user can use.
28
29     struct Token { // struct for the information of a given token
30         bytes32 _tokenId; //Token ID will be created.
31         uint256 timestamp; //Last time device update its proof of live
32     }
33
34     struct Devices { // struct for the addresses of devices
35         address dev;
36         address fog;
37     }
38
39     Token [] public Tokens; // List of all the issued tokens
40     // mapping for users and their accessible devices
41     mapping (address => Devices[]) public users_devices;
42     // mapping for devices at a fog node
43     mapping (address => address[]) public fog_devices;
44
45     modifier onlyOwner { // for user check at modifications
46         //admins.push(msg.sender);
47         //require(msg.sender==admin);
48         bool admin=false;
49         for(uint256 i = 0; i < admins.length; i++){
50             if(msg.sender==admins[i]){
51                 admin=true;
52                 break;
53             }
54         }
55         if(!admin)
56             revert("Not an Admin");
57         _;
58     }
59
60     function No_ofAdmins() public onlyOwner virtual override view returns (uint256) {

```

The interface also shows a list of transactions recorded and a list of deployed contracts. The 'DSCoT AT 0x504...E3544 (BLOCK)' contract is highlighted, showing its balance as 0 ETH and a list of functions available for interaction, such as `addDevice`, `addUser`, `approve`, `delAdmin`, `delDev`, `delUser`, `mintNFT`, `safeTransf...`, `setApprov...`, `transferFr...`, `adminAdd`, `balanceOf`, `fog_devices`, `getApprov...`, `isApprove...`, `No_ofAdm...`, `ownerOf`, `Tokens`, `tokensIss...`, `user_Devi...`, and `users_devi...`.

Figure 18. Accessing the deployed contract