# 277.https://stackoverflow.com/questions/71869681/developing-nft-contract-and-receiveing-external-transactions

**T:**Developing NFT contract and receiveing external transactions

**Q:**I'm developing simple app for nft contracts.  
  
Got stucked at receving external transactions.  
  
Contract:  
  
// SPDX-License-Identifier: MITpragma solidity ^0.8.2;import "@openzeppelin/contracts/token/ERC721/ERC721.sol";import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Burnable.sol";//import "@openzeppelin/contracts/token/ERC721/extensions/IERC721Enumerable.sol";import "@openzeppelin/contracts/access/Ownable.sol";import "@openzeppelin/contracts/utils/Counters.sol";contract RonteNft is ERC721, ERC721URIStorage, ERC721Burnable, Ownable { // IERC721Receiver //IERC721Enumerable using Counters for Counters.Counter; Counters.Counter private \_tokenIdCounter; Counters.Counter private \_soldTokenIdCounter; uint256 public constant BASE\_PRICE = 0.001 ether; struct TokenList { uint256 tokenId; string tokenUri; address tokenOwner; bool sold; } TokenList[] public tokenList; //mapping(bool => TokenList) existingTokenList; address[] public addressList; mapping(address => bool) public WhiteList; constructor() ERC721("RonteNft", "RtNFT") { WhiteList[0x9ACdCfb9385810E0270afB5bFA5000d6Be0891ea] = true; WhiteList[0x2A5e5cc1dA717B1F076Ce04E0c7A814320C0Dc6A] = true; WhiteList[0xEdc64A63ae9dc338052CdF92fA4c80154e4559AE] = true; WhiteList[0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266] = true; WhiteList[0x70997970C51812dc3A010C7d01b50e0d17dc79C8] = true; } function totalSupply() external view returns (uint256){ return \_soldTokenIdCounter.current(); } function safeMint(address to, string memory nftTokenURI) public onlyOwner { \_safeMint(to, \_tokenIdCounter.current()); \_setTokenURI(\_tokenIdCounter.current(), nftTokenURI); tokenList.push( TokenList(\_tokenIdCounter.current(), nftTokenURI, to, false) ); \_tokenIdCounter.increment(); } // The following functions are overrides required by Solidity. function \_burn(uint256 tokenId) internal override(ERC721, ERC721URIStorage) { super.\_burn(tokenId); } function tokenURI(uint256 tokenId) public view override(ERC721, ERC721URIStorage) returns (string memory) { return super.tokenURI(tokenId); } function currentCounter() public view returns (uint256) { return \_tokenIdCounter.current(); } function freeMint(address to, string memory nftTokenURI) public { \_safeMint(to, \_tokenIdCounter.current()); \_setTokenURI(\_tokenIdCounter.current(), nftTokenURI); tokenList.push( TokenList(\_tokenIdCounter.current(), nftTokenURI, to, false) ); \_tokenIdCounter.increment(); } function getTokenList() public view returns (TokenList[] memory) { return tokenList; } function sellToken(address buyer) public { //require(msg.value == BASE\_PRICE, "Incoming amount is wrong!"); require(WhiteList[buyer] == true, "You are not in a white list!"); uint256 tokenId = tokenList[\_soldTokenIdCounter.current()].tokenId; transferFrom(owner(), buyer, tokenId); \_soldTokenIdCounter.increment(); tokenList[\_soldTokenIdCounter.current()].sold = true; } function getError() public { require(1 == 2, "TEST ERROR"); } function deposit() public payable { require(1 == 2, "DEPOSIT ERROR"); } receive() external payable { require(msg.value == BASE\_PRICE, "Incoming amount is wrong!"); require(WhiteList[msg.sender] == true, "You are not in a white list!"); uint256 tokenId = tokenList[\_soldTokenIdCounter.current()].tokenId; approve(msg.sender,tokenId); transferFrom(owner(), msg.sender, tokenId); \_soldTokenIdCounter.increment(); tokenList[\_soldTokenIdCounter.current()].sold = true; } function checkWhiteList(address \_wallet) public returns (bool) { return WhiteList[\_wallet]; } }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Compiling and minting to a local hardhat node(working fine) and then trying to send transaction from another wallet.Receiving error:  
  
 eth\_sendRawTransaction Contract call: RonteNft#<unrecognized-selector> Transaction: 0x5406175da512d812761b0e557ae67240987834294d51f39aa13371ff64453e57 From: 0x70997970c51812dc3a010c7d01b50e0d17dc79c8 To: 0x5fbdb2315678afecb367f032d93f642f64180aa3 Value: 0.001 ETH Gas used: 34976 of 91000 Block #6: 0x12bef472185ef527d650454bb88dd69ad540cc2b5a6dbf3e33c5f928b3472867 Error: VM Exception while processing transaction: reverted with reason string 'ERC721: approve caller is not owner nor approved for all' at RonteNft.approve (@openzeppelin/contracts/token/ERC721/ERC721.sol:116) at RonteNft.<receive> (contracts/RonteNft.sol:112)  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
It's happenging because the sender of transaction is not an owner of a contract, I understand that, but how you can accept payments and transfere token then?If transferFrom can be called only by the owner of the contract?

1 **Answer**

**A1:**There are two methods on ERC721 to achieve this.  
  
You can use approve method;  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
approve(address to, uint256 tokenId)  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
that approves another address to transfer the given token ID The zero address indicates there is no approved address. There can only be one approved address per token at a given time. Can only be called by the token owner or an approved operator.  
  
... or you can use setApprovalForAll method;  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
setApprovalForAll(address to, bool approved)  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
that sets or unsets the approval of a given operator An operator is allowed to transfer all tokens of the sender on their behalf.  
  
You can check the details in here: OpenZeppelin