# 6.https://stackoverflow.com/questions/70770785/internal-json-rpc-error-with-metamask-on-polygon-blockchain-erc721-transfer-c

**T:**Internal JSON-RPC error with MetaMask on Polygon Blockchain. `ERC721: transfer caller is not owner nor approved.`

**Q:**I am making an NFT marketplace. When I deployed my contract on the Mumbai-testnet. The createToken function might work cause it brings up the Metamask for the Gas Fee but after that, the Error occurs something regarding the ONWNERSHIP.(Error image and text is present below.)  
  
STEPS which I follow  
  
 ● npm hardhat node  
  
 ● npm run dev  
  
 ● Selecting the Creating Page.  
  
 ● Enter all the details.  
  
 ● Click on Create an Asset which calls the createToken function.  
  
then the error occurs.  
  
Here is my NFT contract  
  
contract NFT is ERC721URIStorage {using Counters for Counters.Counter;Counters.Counter private \_tokenIds;address contractAddress;constructor(address marketplaceAddress) ERC721("Metaverse Tokens", "METT") { contractAddress = marketplaceAddress;}function createToken(string memory tokenURI) public returns (uint256) { \_tokenIds.increment(); uint256 newItemId = \_tokenIds.current(); \_mint(msg.sender, newItemId); \_setTokenURI(newItemId, tokenURI); setApprovalForAll(contractAddress, true); return newItemId;}}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Here is my NFTMarket contract  
  
contract NFTMarket is ReentrancyGuard {using Counters for Counters.Counter;Counters.Counter private \_itemIds;Counters.Counter private \_itemSold;address payable owner;uint256 listingPrice = 0.025 ether; // Here ether is denoting the MATICconstructor() { owner = payable(msg.sender);}struct MarketItem { uint256 itemId; address nftContract; uint256 tokenId; address payable seller; address payable owner; uint256 price; bool sold;}mapping(uint256 => MarketItem) private idToMarketItem;event MarketItemCreated( uint256 indexed itemId, address indexed nftContract, uint256 indexed tokenId, address seller, address owner, uint256 price, bool sold);function getListingPrice() public view returns (uint256) { return listingPrice;}//Function to create an NFTfunction createMarketItem( address nftContract, uint256 tokenId, uint256 price) public payable nonReentrant { //Conditions for creating the Item. require(price > 0, "Price must be at least 1 wei"); require( msg.value == listingPrice, "Price must be equal to listing price" ); \_itemIds.increment(); uint256 itemId = \_itemIds.current(); idToMarketItem[itemId] = MarketItem( itemId, nftContract, tokenId, payable(msg.sender), payable(address(0)), // When new NFT is created its ownership add is set to 0. price, false ); IERC721(nftContract).transferFrom(msg.sender, address(this), tokenId); //Trigger the Event emit MarketItemCreated( itemId, nftContract, tokenId, msg.sender, address(0), price, false );}//Function to Transfer the Ownershipfunction createMarketSale(address nftContract, uint256 itemId) public payable nonReentrant{ uint256 price = idToMarketItem[itemId].price; uint256 tokenId = idToMarketItem[itemId].tokenId; require( msg.value == price, "Please submit the asking value in order to Purchase" ); //Will transfer the MATIC to the seller address. idToMarketItem[itemId].seller.transfer(msg.value); //Will transfer the ownership from the owner of this contract to the Buyer. IERC721(nftContract).transferFrom(address(this), msg.sender, tokenId); //Set the local value of the owner to the Buyer(msg.sender). idToMarketItem[itemId].owner = payable(msg.sender); //Set this NFT as sold. idToMarketItem[itemId].sold = true; \_itemSold.increment(); payable(owner).transfer(listingPrice);}//Returns number of items unsoldfunction fetchMarketItems() public view returns (MarketItem[] memory) { uint256 itemCount = \_itemIds.current(); uint256 unsoldItemCount = \_itemIds.current() - \_itemSold.current(); uint256 currentIndex = 0; MarketItem[] memory items = new MarketItem[](unsoldItemCount); for (uint256 i = 0; i < itemCount; i++) { if (idToMarketItem[i + 1].owner == address(0)) { uint256 currentId = idToMarketItem[i + 1].itemId; MarketItem storage currentItem = idToMarketItem[currentId]; items[currentIndex] = currentItem; currentIndex += 1; } } return items;}//Returns number of Own(Created or Bought) NFTsfunction fetchMyNFTs() public view returns (MarketItem[] memory) { uint256 totalItemCount = \_itemIds.current(); uint256 itemCount = 0; uint256 currentIndex = 0; for (uint256 i = 0; i < totalItemCount; i++) { if (idToMarketItem[i + 1].owner == msg.sender) { itemCount += 1; } } MarketItem[] memory items = new MarketItem[](itemCount); for (uint256 i = 0; i < totalItemCount; i++) { if (idToMarketItem[i + 1].owner == msg.sender) { uint256 currentId = idToMarketItem[i + 1].itemId; MarketItem storage currentItem = idToMarketItem[currentId]; items[currentIndex] = currentItem; currentIndex += 1; } } return items;}//Returns the no of NFT createdfunction fetchItemsCreated() public view returns (MarketItem[] memory) { uint256 totalItemCount = \_itemIds.current(); uint256 itemCount = 0; uint256 currentIndex = 0; for (uint256 i = 0; i < totalItemCount; i++) { if (idToMarketItem[i + 1].seller == msg.sender) { itemCount += 1; } } MarketItem[] memory items = new MarketItem[](itemCount); for (uint256 i = 0; i < totalItemCount; i++) { if (idToMarketItem[i + 1].seller == msg.sender) { uint256 currentId = idToMarketItem[i + 1].itemId; MarketItem storage currentItem = idToMarketItem[currentId]; items[currentIndex] = currentItem; currentIndex += 1; } } return items;}}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
I tried changing the RPC in the MetaMask and the configuration files and redeployed it many times with different accounts, but still, nothing changes.  
  
The Error  
  
 MetaMask - RPC Error: Internal JSON-RPC error. data:code: 3message: "execution reverted: ERC721: transfer caller is not owner nor approved"  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Image of the console  
  
If any other info is required please comment  
  
Link of Blockchain Explorer

**C1:**Can you link the failing transaction on a blockchain explorer?

**C2:**Done sir, edited. In my opinion it create the token id but the approval is not working.

**C3:**Thanks for the link. However the linked transaction is not failing. Can you provide steps to reproduce the error message (such as what function, from which address, which token ID, ...)?

**C4:**Sir, I have updated the Query and put the image of the console there. I hope this may provide you the complete info. If anything left please let me know.

3 **Answer**

**A1:**Thank You for your efforts.I got the solution to it(After Searching for 3 days).  
  
SOLUTION=>  
  
There's currently a bug on Mumbai causing deployed addresses to be incorrect. This is causing the constructor of the NFT contract to approve the wrong address for NFT purchases (because it uses the address of the Market deployment for approval) — causing the annoying "execution reverted: ERC721: approve caller is not owner nor approved for all" error.  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Try using Mainnet (yes, you'll have to use real money) but it works!  
  
Reference  
  
Here's a workaround deploy script that will make it work on Mumbai. Replace main() in deploy.js with:  
  
const hre = require("hardhat");async function main() { const [deployer] = await hre.ethers.getSigners(); console.log( "Deploying contracts with the account:", deployer.address ); let txHash, txReceipt const NFTMarket = await hre.ethers.getContractFactory("NFTMarket"); const nftMarket = await NFTMarket.deploy(); await nftMarket.deployed(); txHash = nftMarket.deployTransaction.hash; txReceipt = await ethers.provider.waitForTransaction(txHash); let nftMarketAddress = txReceipt.contractAddress console.log("nftMarket deployed to:", nftMarketAddress); const NFT = await hre.ethers.getContractFactory("NFT"); const nft = await NFT.deploy(nftMarketAddress); await nft.deployed(); txHash = nft.deployTransaction.hash; // console.log(`NFT hash: ${txHash}\nWaiting for transaction to be mined...`); txReceipt = await ethers.provider.waitForTransaction(txHash); let nftAddress = txReceipt.contractAddress console.log("nft deployed to:", nftAddress);}main() .then(() => process.exit(0)) .catch((error) => { console.error(error); process.exit(1); });  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Redeploy the Contracts with this Script, and change the config.js.  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**A2:**I checked your full code and it is working.  
  
You are inheriting from ERC721URIStorage which inherits from ERC721 If you check the transferFrom inside ERC721:  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
function transferFrom(address from,address to,uint256 tokenId ) public virtual override { // \*\*\*\*\* THIS REQUIRE IS NOT SATISFIED \*\*\*\*\* require(\_isApprovedOrOwner(\_msgSender(), tokenId), "ERC721: transfer caller is not owner nor approved"); \_transfer(from, to, tokenId); }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
you are getting that error, because require statement inside transferFrom is not satisfied.  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
this is the \_isApprovedOrOwner  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
 function \_isApprovedOrOwner(address spender, uint256 tokenId) internal view virtual returns (bool) { require(\_exists(tokenId), "ERC721: operator query for nonexistent token"); address owner = ERC721.ownerOf(tokenId); return (spender == owner || getApproved(tokenId) == spender || isApprovedForAll(owner, spender)); }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
this function is not returning True. in order to get True, this  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
spender == owner || getApproved(tokenId) == spender || isApprovedForAll(owner, spender)   
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
should return True. in or operations in order to get True, 3 of conditions must be True.  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
In my opinion, you are trying to transfer a token that is not yours.

**C1:**I did use the second statement to transfer from. I have added my MARKET CONTRACT please have a look. And if any other info require let me know.

**C2:**when do u get the error. while you compile or you are calling from frontend

**C3:**When I call from the frontend.

**C4:**I have step by step checked the code 4 5 times with the Youtube I have practising from. There is no difference in code, still I am getting this unknown issue.

**C5:**share the front-end code please. your contract code looks ok

**A3:**During development, every time you compile your smart contract (using something like hardhat compile..) , you have to copy the new smart contract addresses generated in your command prompt and paste them in the designated location in your source code.  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]