# 606.https://stackoverflow.com/questions/69366331/why-am-i-receiving-creation-of-nft-errored-internal-json-rpc-error-code

**T:**Why am I receiving "creation of NFT errored: Internal JSON-RPC error. { "code": -32000, "message": "execution reverted" }" in my remix.eth. code?

**Q:**I am trying to publish my collection of 5,000 NFTs and I'm just about to upload them to Opensea... I just keep receiving the error in the title. I had no problem producing 1, 20, or 50 NFTs of the 5000. But when I changed the number to 5000, for some reason I get this error. Does anyone know what's wrong? Appreciate any feedback :)  
  
pragma solidity >=0.7.0 <0.9.0;import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol";import "@openzeppelin/contracts/access/Ownable.sol";contract NFT is ERC721Enumerable, Ownable { using Strings for uint256; string public baseURI; string public baseExtension = ".json"; uint256 public cost = 0.05 ether; uint256 public maxSupply = 10000; uint256 public maxMintAmount = 5000; bool public paused = false; mapping(address => bool) public whitelisted; constructor( string memory \_name, string memory \_symbol, string memory \_initBaseURI ) ERC721(\_name, \_symbol) { setBaseURI(\_initBaseURI); mint(msg.sender, 5000); } // internal function \_baseURI() internal view virtual override returns (string memory) { return baseURI; } // public function mint(address \_to, uint256 \_mintAmount) public payable { uint256 supply = totalSupply(); require(!paused); require(\_mintAmount > 0); require(\_mintAmount <= maxMintAmount); require(supply + \_mintAmount <= maxSupply); if (msg.sender != owner()) { if(whitelisted[msg.sender] != true) { require(msg.value >= cost \* \_mintAmount); } } for (uint256 i = 1; i <= \_mintAmount; i++) { \_safeMint(\_to, supply + i); } } function walletOfOwner(address \_owner) public view returns (uint256[] memory) { uint256 ownerTokenCount = balanceOf(\_owner); uint256[] memory tokenIds = new uint256[](ownerTokenCount); for (uint256 i; i < ownerTokenCount; i++) { tokenIds[i] = tokenOfOwnerByIndex(\_owner, i); } return tokenIds; } function tokenURI(uint256 tokenId) public view virtual override returns (string memory) { require( \_exists(tokenId), "ERC721Metadata: URI query for nonexistent token" ); string memory currentBaseURI = \_baseURI(); return bytes(currentBaseURI).length > 0 ? string(abi.encodePacked(currentBaseURI, tokenId.toString(), baseExtension)) : ""; } //only owner function setCost(uint256 \_newCost) public onlyOwner() { cost = \_newCost; } function setmaxMintAmount(uint256 \_newmaxMintAmount) public onlyOwner() { maxMintAmount = \_newmaxMintAmount; } function setBaseURI(string memory \_newBaseURI) public onlyOwner { baseURI = \_newBaseURI; } function setBaseExtension(string memory \_newBaseExtension) public onlyOwner { baseExtension = \_newBaseExtension; } function pause(bool \_state) public onlyOwner { paused = \_state; } function whitelistUser(address \_user) public onlyOwner { whitelisted[\_user] = true; } function removeWhitelistUser(address \_user) public onlyOwner { whitelisted[\_user] = false; } function withdraw() public payable onlyOwner { (bool success, ) = payable(msg.sender).call{value: address(this).balance}(""); require(success); }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

0 **Answer**