# 617.https://stackoverflow.com/questions/69239240/polygon-transaction-working-just-fine-on-mumbai-but-not-on-mainnet

**T:**Polygon transaction working just fine on Mumbai but not on Mainnet

**Q:**Hello I'm trying to mint an NFT using Polygon and it works just fine on Mumbai but as soon as i switch over to the mainnet the transaction doesn't go through instead of going through in 5 seconds on mumbai. Even though im using the exact same contract just deployed on the mainnet instead of Mumbai and the code is the same too. All im doing is switching the contract address and rpc url but for some reason it just doesn't work on the Polygon mainnet below is the code im using.  
  
// Init contract const contractABI = require('../../contract-abi.json'); const contractAddress = config.mintingContractAddress; const contract = await new this.web3.eth.Contract(contractABI, contractAddress); // Mint NFT const nft = contract.methods.mintNFT(user.walletAddress, metadataUploadURL, user.paymentAddress).encodeABI(); // Get gas pricing const priorityFees = await axios.get('https://gasstation-mainnet.matic.network'); const estBaseGas = await this.web3.eth.estimateGas({ data: nft, to: contractAddress, }); console.log('USING GAS: ' + estBaseGas); // Sign NFT minting transaction const totalGas = estBaseGas + priorityFees.data.standard; console.log('TOTALGAS: ', Math.round(totalGas).toString()); const transaction = await this.web3.eth.accounts.signTransaction( { from: user.walletAddress, to: contractAddress, nonce: await this.web3.eth.getTransactionCount(user.walletAddress, 'pending'), // Get count of all transactions sent to the contract from this address including pending ones data: nft, // maxPriorityFee: priorityFees.data.average, Not supported on Polygon MATIC yet gas: Math.round(totalGas).toString(), gasPrice: await this.web3.eth.getGasPrice(), }, wallet.privateKey, ); this.logger.silly('Finished signing NFT transaction'); // Send the transaction that we signed const mintT = await this.web3.eth.sendSignedTransaction(transaction.rawTransaction); this.logger.silly('Sent transaction'); console.log(mintT);  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Also tried this for signing  
  
// Get gas pricing const priorityFees = await axios.get('https://gasstation-mainnet.matic.network'); const estBaseGas = await this.web3.eth.estimateGas({ data: nft, to: contractAddress, }); console.log('USING GAS: ' + estBaseGas); // Sign NFT minting transaction const totalGas = estBaseGas + priorityFees.data.standard; console.log('TOTALGAS: ', Math.round(totalGas).toString()); console.log('P', priorityFees.data.standard); const gp = this.web3.utils.toWei(priorityFees.data.standard.toString(), 'Gwei').toString(); console.log('GP', gp); const transaction = await this.web3.eth.accounts.signTransaction( { from: user.walletAddress, to: contractAddress, nonce: await this.web3.eth.getTransactionCount(user.walletAddress, 'pending'), // Get count of all transactions sent to the contract from this address including pending ones data: nft, // maxPriorityFee: priorityFees.data.average, Not supported on Polygon MATIC yet gas: '1000000', gasPrice: gp, }, wallet.privateKey, );  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Mempool explorer for transaction that takes forever and nearly instant one.Forever:Instant:One on mainnet that used 30 gwei of gas:Does anybody know why this is happening?Also yes i do know that the fast one does have 2 extra gwei in gas but even setting it to that manually it still takes forever and according to https://polygonscan.com/gastracker even with one gwei it should be processed within 30 seconds. Even when using 50 Gwei it seems to take hours to process or maybe it's being dropped? The transactions don't even seem to be getting to the contract they are just stuck somewhere in the chain.contract address: 0xa915E82285e6F82eD10b0579511F48fD716a2043  
  
contract source code:  
  
//SPDX-License-Identifier: UNLICENSEDpragma solidity ^0.8.0;import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";import "@openzeppelin/contracts/utils/Counters.sol";contract MyNFT is ERC721URIStorage { using Counters for Counters.Counter; Counters.Counter private \_tokenIds; event MintedNFT(address recipent,string tokenURI,address artist, uint256 tokenID); mapping(uint256 => address) private artists; // Used to store token ids => artist addresses // mapping(uint256 => uint256) private royalties; // tokenId => royaltyPercentage // mapping(uint256 => address) private nftMintInitators; // Used to store token ids => sender addresses // mapping(uint256 => bool) private royaltiesSet; constructor(string memory name\_, string memory symbol\_) ERC721(name\_, symbol\_) { } // // Support for https://eips.ethereum.org/EIPS/eip-2981 // /// @notice Called with the sale price to determine how much royalty // // is owed and to whom. // /// @param \_tokenId - the NFT asset queried for royalty information // /// @param \_salePrice - the sale price of the NFT asset specified by \_tokenId // /// @return receiver - address of who should be sent the royalty payment // /// @return royaltyAmount - the royalty payment amount for \_salePrice // function royaltyInfo( // uint256 \_tokenId, // uint256 \_salePrice // ) external view returns ( // address receiver, // uint256 royaltyAmount // ) { // return ( // artists[\_tokenId], // \_salePrice \* royalties[\_tokenId] // Take percentage // ); // } // function updateRoyaltyPercentage( // uint256 royaltyPercentage, // In decimal like 0.5 or 0.25 (Send 0.0 for no royalties) // uint256 tokenID // ) public { // if (msg.sender == nftMintInitators[tokenID] && royaltiesSet[tokenID] == false) { // royalties[tokenID] = royaltyPercentage; // royaltiesSet[tokenID] = true; // } // } function mintNFT(address recipient, string memory tokenURI, address artist // Address for the artist not using \_msgSender() because this transaction is sent by the users NFT holding account ) public returns (uint256) { \_tokenIds.increment(); uint256 newItemId = \_tokenIds.current(); \_mint(recipient, newItemId); \_setTokenURI(newItemId, tokenURI); artists[newItemId] = artist; // nftMintInitators[newItemId] = msg.sender; // royaltiesSet[newItemId] = false; emit MintedNFT(recipient,tokenURI,artist,newItemId); return newItemId; }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**C1:**Please edit your question and share the contract source code (incl. the constructor argument values if there are any) or verify it on the blockchain explorer. It's possible that the mintNFT() fails on the mainnet because of some dependency call (that might go through on the testnet but fails on the mainnet).

**C2:**@PetrHejda ok added

1 **Answer**

**A1:**You can test with simple code just to mint one NFT. Adding gasPrice and gasLimit parameters directly into minting function could help  
  
 const hre = require("hardhat"); async function main() { const NFT = await hre.ethers.getContractFactory("NFTNAME"); const WALLET\_ADDRESS = "0xxxxxxxxxxxxxx" const CONTRACT\_ADDRESS = "0xa915E82285e6F82eD10b0579511F48fD716a2043" const contract = NFT.attach(CONTRACT\_ADDRESS); mintedNFT = await contract.mintNFT(WALLET\_ADDRESS,{ gasLimit: 285000, gasPrice: ethers.utils.parseUnits('30', 'gwei')}); console.log("NFT minted:", mintedNFT);}main().then(() => process.exit(0)).catch(error => { console.error(error); process.exit(1);});  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]