# 8.https://stackoverflow.com/questions/71678110/how-can-i-set-2-max-supplys-in-a-erc721

**T:**How can i set 2 max supply's in a ERC721

**Q:**I have a problem. The problem is that i want a max mint supply for the whitelist sale and public sale. So for example;In total i have 10.000 NFT's. There will be an whitelist sale and a public sale. For the whitelist sale i have 2.000 wallet addresses, but i only want them to be able to mint 1.500 NFT's. and in the public sale i want the remaining 8.500 NFT's to be sold.  
  
I already tried somethings with the normal maxsupply but can't figure it out. I want a max mint per wallet like 10 and cant seem to limit the supply to 1500 for only the whitelist  
  
Does anyone can explain this to me or have a code example?

2 **Answer**

**A1:**I don't recommend limiting your whitelist per-wallet minting amount (you want to mint out, don't you?!), but I understand the reasons why you might. I'll provide both options. All code is abbreviated for brevity sake:  
  
\*Note 1: We will be using OpenZeppelin Utilities - Counters for tracking minted progress. You can also consider using totalSupply(), however, if burned tokens are a concern totalSupply() will decrement and throw off your count, whereas Counters will not.  
  
Note 2: This assumes you're whitelist occurs before public and that you're not also juggling a reserve count as well - additional checks and counters would be required for that.  
  
Note 3: This covers ONLY the check for limiting whitelist; you will obviously also need additional checks for valid whitelist account, sufficient payment, etc.  
  
Limit Whitelist Total Supply  
  
...import "@openzeppelin/contracts/utils/Counters.sol";...error ExceededWhitelistSupply();...using Counters for Counters.Counter;uint256 public maxSupply = 10000;uint256 public maxWhitelistSupply = 1500;Counters.Counter private totalWhitelistSupply;...function mintWhitelist(uint256 \_qty) external payable { if ( totalWhitelistSupply.current() + \_qty > maxWhitelistSupply ) revert ExceededWhitelistSupply(); for (uint256 i = 0; i < \_qty; i++) { totalWhitelistSupply.increment(); } \_mint(msg.sender, \_qty, '', true);}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Limit Wallet && Limit Whitelist Total Supply  
  
...import "@openzeppelin/contracts/utils/Counters.sol";...error ExceededWhitelistSupply();error ExceededMaxPerWallet();...using Counters for Counters.Counter;uint256 public maxSupply = 10000;uint256 public maxWhitelistSupply = 1500;uint256 public maxWhitelistPerWallet = 10;Counters.Counter private totalWhitelistSupply;mapping(address => uint256) public whitelistMintedAmount;...function mintWhitelist(uint256 \_qty) external payable { if ( whitelistMintedAmount[msg.sender] + \_qty > maxWhitelistPerWallet ) revert ExceededMaxPerWallet(); if ( totalWhitelistSupply.current() + \_qty > maxWhitelistSupply ) revert ExceededWhitelistSupply(); for (uint256 i = 0; i < \_qty; i++) { totalWhitelistSupply.increment(); } whitelistMintedAmount[msg.sender] += \_qty; \_mint(msg.sender, \_qty, '', true);}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Here, we've used mapping - good tutorial here - to track the number of NFTs that have been minted to this wallet (preferred method, as this won't be fooled by the account transferring NFTs out of the wallet and then minting more). If you want to go WAY down the rabbit hole, you can also look at trash-canning this whole approach and learn up on this approach for handling your whitelist.  
  
Keep in mind that there are more checks and balances that you'll need to add (e.g., visualizing this in the front end of your dApp to avoid minting when they shouldn't be able to, additional validation layers in your mint functions, etc.), but this should provide you with the core pieces needed for limiting by a max wallet and max supply. I apologize for any code errors - this is my first StackOverflow answer and the short-handing and readability of the code is a bit difficult to error check.

**A2:** ● Have a whitelistMaxSupply  
  
 ● Have a getMaxSupply function  
  
 ● Have a finishWhitelist function  
  
 ● Have a maxSupply  
  
Unless whitelist finished, getMaxSupply will return whitelistMaxSupply, after finished, it will return maxSupply  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
 ● Profit  
  
There are many ways to solve your problem, this is just the first one that came to my mind  
  
function getMaxSupply() view public returns(uint256){ if(whitelistFinished){ return maxSupply; } return whitelistMaxSupply;}function finishWhitelist() public{ whitelistFinished = true;}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**C1:**So when finishwhitelist, the public will automaticly start?

**C2:**let me give you an example @JamesL

**C3:**I can’t figure it out man could we get on a call or some? Or how can we fix this?

**C4:**@JamesL do you use telegram?

**C5:**Yeah, i will add you. whats your telegram