# 186.https://stackoverflow.com/questions/72484886/how-to-get-the-candy-machine-id-using-a-nft-created-by-the-candy-machine

**T:**How to get the candy machine ID using a NFT created by the candy machine?

**Q:**Suppose I only have the mint address of a single NFT created by a specific candy machine.How can I use the mint address and ultimately get the candy machine ID? is it even possible?

2 **Answer**

**A1:**A fast way to get the CMid using an NFT is fetching the first tx that the NFT has (the oldest one) and checking the fifth instruction, then the first account on this instruction is the Candy Machine used to create and mint the NFT.  
  
For example lets take this NFT 3GXHJJd1DfEn1PVip87uUJXjeW1jDgeJb3B7a6xHWAeJ, the oldest transaction that is has is this one. Then you can see on the image below that the first account on the 5th instruction is: H2oYLkXdkX38eQ6VTqs26KAWAvEpYEiCtLt4knEUJxpu (Note that this CM account is empty because they withdraw and close the account after mint).  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
  
  
You can do it using some explorer of with code using solana/web3.js

**C1:**Nice thank you for the response! Is there a way to get the first trx quickly ?

**A2:**as per the official documentation:https://docs.metaplex.com/guides/mint-lists  
  
The typical method to create the mint list is to a use a tool that finds all NFTs with a specific creator in the first position of the creators array. If your NFTs were minted with a candy machine this will be the candy machine creator id by default. If you have multiple candy machines that are part of the collection, you can create a separate mint list for each candy machine and combine them together to create a single mint list which you provide to the marketplace(s) you are listing with.  
  
And how to get the creators from a mint address is by getting the metadata associated with the mint address.  
  
The metadata is encoded in a specific format for which you can use the metaplex libraries to decode.Here is a simple python example: https://github.com/michaelhly/solana-py/issues/48#issuecomment-1073077165  
  
def unpack\_metadata\_account(data): assert(data[0] == 4) i = 1 source\_account = base58.b58encode(bytes(struct.unpack('<' + "B"\*32, data[i:i+32]))) i += 32 mint\_account = base58.b58encode(bytes(struct.unpack('<' + "B"\*32, data[i:i+32]))) i += 32 name\_len = struct.unpack('<I', data[i:i+4])[0] i += 4 name = struct.unpack('<' + "B"\*name\_len, data[i:i+name\_len]) i += name\_len symbol\_len = struct.unpack('<I', data[i:i+4])[0] i += 4 symbol = struct.unpack('<' + "B"\*symbol\_len, data[i:i+symbol\_len]) i += symbol\_len uri\_len = struct.unpack('<I', data[i:i+4])[0] i += 4 uri = struct.unpack('<' + "B"\*uri\_len, data[i:i+uri\_len]) i += uri\_len fee = struct.unpack('<h', data[i:i+2])[0] i += 2 has\_creator = data[i] i += 1 creators = [] verified = [] share = [] if has\_creator: creator\_len = struct.unpack('<I', data[i:i+4])[0] i += 4 for \_ in range(creator\_len): creator = base58.b58encode(bytes(struct.unpack('<' + "B"\*32, data[i:i+32]))) creators.append(creator) i += 32 verified.append(data[i]) i += 1 share.append(data[i]) i += 1 primary\_sale\_happened = bool(data[i]) i += 1 is\_mutable = bool(data[i]) metadata = { "update\_authority": source\_account, "mint": mint\_account, "data": { "name": bytes(name).decode("utf-8").strip("\x00"), "symbol": bytes(symbol).decode("utf-8").strip("\x00"), "uri": bytes(uri).decode("utf-8").strip("\x00"), "seller\_fee\_basis\_points": fee, "creators": creators, "verified": verified, "share": share, }, "primary\_sale\_happened": primary\_sale\_happened, "is\_mutable": is\_mutable, } return metadata  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
and  
  
METADATA\_PROGRAM\_ID = PublicKey('metaqbxxUerdq28cj1RbAWkYQm3ybzjb6a8bt518x1s')def get\_nft\_pda(mint\_key): return(PublicKey.find\_program\_address([b'metadata', bytes(METADATA\_PROGRAM\_ID), bytes(PublicKey(mint\_key))],METADATA\_PROGRAM\_ID)[0])def get\_metadata(mint\_key): data = base64.b64decode(solana\_client.get\_account\_info(get\_nft\_pda(mint\_key))['result']['value']['data'][0]) return(unpack\_metadata\_account(data))  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
An example getting Candy Machine ID using this method for the "Aw4RhpcW5rod2Afhp7dhv2XrMZyNJpzVdYkjJ7kkYzpS" mint address would result in:  
  
 "update\_authority": "DGNZDSvy6emDXvBuCDRrpLVxcPaEcvKiStvvCivEJ38X", "mint": "Aw4RhpcW5rod2Afhp7dhv2XrMZyNJpzVdYkjJ7kkYzpS", "data": { "name": "Shadowy Super Coder #5240", "symbol": "SSC", "uri": "https://shdw-drive.genesysgo.net/8yHTE5Cz3hwcTdghynB2jgLuvKyRgKEz2n5XvSiXQabG/5240.json", "seller\_fee\_basis\_points": 500, "creators": [ "71ghWqucipW661X4ht61qvmc3xKQGMBGZxwSDmZrYQmf", "D6wZ5U9onMC578mrKMp5PZtfyc5262426qKsYJW7nT3p" ], "verified": [ 1, 0 ], "share": [ 0, 100 ] }, "primary\_sale\_happened": true, "is\_mutable": true}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
In this case, for the Collection SSC, the candy machine ID is 71ghWqucipW661X4ht61qvmc3xKQGMBGZxwSDmZrYQmf