# 520.https://stackoverflow.com/questions/70597753/how-to-find-all-nfts-minted-from-a-v2-candy-machine

**T:**How to find all NFTs minted from a v2 candy machine

**Q:**I'm minting Solana NFTs. Candy machine v2 was recently released and v1 is deprecated.  
  
If I create a v2 candy machine and mint some NFTs, how can I later find the hashes from all the tokens that were minted?

1 **Answer**

**A1:**If you have the candy machine id, you can find all mints with this:  
  
import { Connection } from '@metaplex/js';import { Metadata, MetadataProgram } from '@metaplex-foundation/mpl-token-metadata';const connection = new Connection('mainnet-beta');const MAX\_NAME\_LENGTH = 32;const MAX\_URI\_LENGTH = 200;const MAX\_SYMBOL\_LENGTH = 10;const MAX\_CREATOR\_LEN = 32 + 1 + 1;const candyMachineId: string = 'BdNtsrV26ZHdqDFxmDfLib6CrcUNj4ePorhppHreRgER';export async function fetchHashTable(hash: string){ const metadataAccounts = await MetadataProgram.getProgramAccounts( connection, { filters: [ { memcmp: { offset: 1 + 32 + 32 + 4 + MAX\_NAME\_LENGTH + 4 + MAX\_URI\_LENGTH + 4 + MAX\_SYMBOL\_LENGTH + 2 + 1 + 4 + 0 \* MAX\_CREATOR\_LEN, bytes: hash, }, }, ], }, ) const mintHashes: any = [] for (let index = 0; index < metadataAccounts.length; index++) { const account = metadataAccounts[index]; const accountInfo: any = await connection.getParsedAccountInfo(account.pubkey); const metadata = new Metadata(hash.toString(), accountInfo.value); mintHashes.push(metadata.data.mint) } console.log(mintHashes)}fetchHashTable(candyMachineId)  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
You can find this and more on the solana cookbook  
  
EDIT: The above was for Candy Machine V1.  
  
For Candy Machine V2, you would do the following:  
  
import { Connection, clusterApiUrl, PublicKey } from '@solana/web3.js';import bs58 from 'bs58';const connection = new Connection(clusterApiUrl('mainnet-beta'));const MAX\_NAME\_LENGTH = 32;const MAX\_URI\_LENGTH = 200;const MAX\_SYMBOL\_LENGTH = 10;const MAX\_CREATOR\_LEN = 32 + 1 + 1;const MAX\_CREATOR\_LIMIT = 5;const MAX\_DATA\_SIZE = 4 + MAX\_NAME\_LENGTH + 4 + MAX\_SYMBOL\_LENGTH + 4 + MAX\_URI\_LENGTH + 2 + 1 + 4 + MAX\_CREATOR\_LIMIT \* MAX\_CREATOR\_LEN;const MAX\_METADATA\_LEN = 1 + 32 + 32 + MAX\_DATA\_SIZE + 1 + 1 + 9 + 172;const CREATOR\_ARRAY\_START = 1 + 32 + 32 + 4 + MAX\_NAME\_LENGTH + 4 + MAX\_URI\_LENGTH + 4 + MAX\_SYMBOL\_LENGTH + 2 + 1 + 4;const TOKEN\_METADATA\_PROGRAM = new PublicKey('metaqbxxUerdq28cj1RbAWkYQm3ybzjb6a8bt518x1s');const CANDY\_MACHINE\_V2\_PROGRAM = new PublicKey('cndy3Z4yapfJBmL3ShUp5exZKqR3z33thTzeNMm2gRZ');const candyMachineId = new PublicKey('ENTER\_YOUR\_CANDY\_MACHINE\_ID\_HERE');const getMintAddresses = async (firstCreatorAddress: PublicKey) => { const metadataAccounts = await connection.getProgramAccounts( TOKEN\_METADATA\_PROGRAM, { // The mint address is located at byte 33 and lasts for 32 bytes. dataSlice: { offset: 33, length: 32 }, filters: [ // Only get Metadata accounts. { dataSize: MAX\_METADATA\_LEN }, // Filter using the first creator. { memcmp: { offset: CREATOR\_ARRAY\_START, bytes: firstCreatorAddress.toBase58(), }, }, ], }, ); return metadataAccounts.map((metadataAccountInfo) => ( bs58.encode(metadataAccountInfo.account.data) ));};const getCandyMachineCreator = async (candyMachine: PublicKey): Promise<[PublicKey, number]> => ( PublicKey.findProgramAddress( [Buffer.from('candy\_machine'), candyMachine.toBuffer()], CANDY\_MACHINE\_V2\_PROGRAM, ));(async () => { const candyMachineCreator = await getCandyMachineCreator(candyMachineId); getMintAddresses(candyMachineCreator[0]);})();  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Make sure you replace ENTER\_YOUR\_CANDY\_MACHINE\_ID\_HERE with your candy machine id  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**C1:**This looks great!! Thanks so much. If you have time, can you explain this code a bit? In particular, what's happening in filters.memcmp.offset? I don't see that in the cookbook. I'll test this tonight and add a bounty tomorrow.

**C2:**@emersonthis There's a good overview on how to use those params in the cookbook. Ultimately it is helping you find the right offset for what you are looking for in the accounts.

**C3:**I'm not getting any results when I run this code. I npm ied the two packages and replaced candyMachineId with the address of my candymachine. But when I run the script with ts-node I get an empty array [ ]. I know the minting worked because I see all the NFTs in my wallet. Any ideas what I'm doing wrong?

**C4:**I have a hunch: I minted using candy machine v2... is this script incompatible with the newest release?

**C5:**I keep getting a timeout on multiple different endpoints when running getMintAdresses