# 321.https://stackoverflow.com/questions/71640223/interact-with-solana-program-directly

**T:**Interact with solana program directly

**Q:**I'm trying to use solana-py to interact with a solana program where i don't have documentation about.  
  
The program is the Solana Magic Eden NFT Marketplace .I already tried to get information about it from anchor but it has no data available.  
  
My goal: I would like to list a NFT directly without the web-interface.  
  
My test: As an example, i would like to list this NFT which i own: Robber#04977.  
  
As i don't know anything about the program as their is no documentation available i looked into other transaction and found one which i tried to recreate but with my nft: Successful transaction.  
  
I changed the Accounts, used my private key and created the transaction using solana-py: my failed transaction.  
  
Please see edit below for current state.  
  
Original code:  
  
from solana.transaction import AccountMeta, Transaction, TransactionInstructionfrom solana.rpc.types import TxOptsfrom solana.account import Accountfrom solana.rpc.api import Clientfrom solana.publickey import PublicKeyfrom solana.keypair import Keypairfrom getpass import getpassimport base58# setup clienturl = 'https://api.mainnet-beta.solana.com'client = Client(url)program = 'MEisE1HzehtrDpAAT8PnLHjpSSkRYakotTuJRPjTpo8'# get account from private keypwd = getpass('Chrome -> Phantom -> Settings -> Export private Key')byte\_array = base58.b58decode(pwd)keypair = list(map(lambda b: int(str(b)), byte\_array))[:]initializer\_account = Keypair(keypair[0:32])# create transaction and sign ittxn = Transaction(recent\_blockhash=client.get\_recent\_blockhash()['result']['value']['blockhash'], fee\_payer=initializer\_account.public\_key)txn.add( TransactionInstruction( keys=[ AccountMeta(pubkey=PublicKey(initializer\_account.public\_key), is\_signer=True, is\_writable=True), AccountMeta(pubkey=PublicKey('GG24iCpytsz2nxei81FHyEyduQAxCAJHWkDLitwr9MxQ'), is\_signer=False, is\_writable=True), AccountMeta(pubkey=PublicKey('3gS9AqTJ9adw23tZ87Hn1ccyYJ5KZ5tcoNQfYhCFu2e3'), is\_signer=False, is\_writable=True), AccountMeta(pubkey=PublicKey('TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA'), is\_signer=False, is\_writable=False), AccountMeta(pubkey=PublicKey('11111111111111111111111111111111'), is\_signer=False, is\_writable=False), ], program\_id=PublicKey('MEisE1HzehtrDpAAT8PnLHjpSSkRYakotTuJRPjTpo8'), data=bytes.fromhex('96d480ba740183710094357700000000ff') # sell für 2 Solana ))txn.sign(initializer\_account)rpc\_response = client.send\_transaction( txn, initializer\_account, opts=TxOpts(skip\_preflight=True, skip\_confirmation=False))  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
The response contains the transaction id to my failed transaction.  
  
{'jsonrpc': '2.0', 'result': 'NytmsBK59kckV3nGBsw6Vi9XAw8jkpkQGgHKCMYNFPYXLy57caNN7icNpMepofNsdncJ2BVziFJ82e8PKpH1EnV', 'id': 3}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
The program log from solscan looks like this:  
  
#1 Magic Eden NFT Marketplace instruction> Program Magic Eden NFT Marketplace consumed 5829 of 200000 compute units> Program returned error: Could not create program address with signer seeds: Provided seeds do not result in a valid address  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Maybe it is some wrong data which i'm sending to the program. I just looked for historic transactions which were successful and used it for my transaction too.  
  
edit: getting closer, but not yet done.I'm now also creating the instructions to create a account and to set authority. But these are done serial and not as an inner instruction.  
  
from solana.transaction import AccountMeta, Transaction, TransactionInstructionfrom solana.rpc.types import TxOptsfrom solana.account import Accountfrom solana.rpc.api import Clientfrom solana.publickey import PublicKeyfrom solana.rpc.commitment import Recent, Rootfrom solana.keypair import Keypairfrom solana.system\_program import create\_account, CreateAccountParamsfrom getpass import getpassimport base58from spl.token.instructions import set\_authority, SetAuthorityParams, AuthorityTypeLAMPORTS\_PER\_SOL = 1000000000url = 'https://api.mainnet-beta.solana.com'client = Client(url)pwd = getpass('Chrome -> Phantom -> Settings -> Export private Key')# setup of accountsprogram = 'MEisE1HzehtrDpAAT8PnLHjpSSkRYakotTuJRPjTpo8'# get int based keypair of accountbyte\_array = base58.b58decode(pwd)keypair = list(map(lambda b: int(str(b)), byte\_array))[:]initializer\_account = Keypair(keypair[0:32])token\_account\_robber = PublicKey('GG24iCpytsz2nxei81FHyEyduQAxCAJHWkDLitwr9MxQ')# inner instruction: create accountfrom\_account, new\_account = initializer\_account.public\_key, Keypair().public\_keyinner\_instruction1 = create\_account( CreateAccountParams( from\_pubkey=from\_account, new\_account\_pubkey=new\_account, lamports=int(0.00144768\*LAMPORTS\_PER\_SOL), space=1, program\_id=PublicKey('MEisE1HzehtrDpAAT8PnLHjpSSkRYakotTuJRPjTpo8')))# make all accounts of this non signer and non writeableinner\_instruction1.keys[0].is\_signer=Falseinner\_instruction1.keys[0].is\_writable=Falseinner\_instruction1.keys[1].is\_signer=Falseinner\_instruction1.keys[1].is\_writable=False# inner instruction: set authorityinner\_instruction2 = set\_authority( SetAuthorityParams( program\_id=PublicKey('TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA'), account=token\_account\_robber, authority=AuthorityType.ACCOUNT\_OWNER, current\_authority=initializer\_account.public\_key, new\_authority=PublicKey('GUfCR9mK6azb9vcpsxgXyj7XRPAKJd4KMHTTVvtncGgp') ))# combine all instructions txn = Transaction(recent\_blockhash=client.get\_recent\_blockhash()['result']['value']['blockhash'], fee\_payer=initializer\_account.public\_key)txn.add( TransactionInstruction( keys=[ AccountMeta(pubkey=PublicKey(initializer\_account.public\_key), is\_signer=True, is\_writable=True), AccountMeta(pubkey=PublicKey('GG24iCpytsz2nxei81FHyEyduQAxCAJHWkDLitwr9MxQ'), is\_signer=False, is\_writable=True), AccountMeta(pubkey=new\_account, is\_signer=False, is\_writable=True), AccountMeta(pubkey=PublicKey('TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA'), is\_signer=False, is\_writable=False), AccountMeta(pubkey=PublicKey('11111111111111111111111111111111'), is\_signer=False, is\_writable=False), ], program\_id=PublicKey('MEisE1HzehtrDpAAT8PnLHjpSSkRYakotTuJRPjTpo8'), data=bytes.fromhex('96d480ba740183710094357700000000ff') # sell für 2 Solana ))txn.add(inner\_instruction1)txn.add(inner\_instruction2)# sign and sendtxn.sign(initializer\_account)rpc\_response = client.send\_transaction( txn, initializer\_account, opts=TxOpts(skip\_preflight=True, skip\_confirmation=False))  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
The 'rpc\_reponse` contains the following:  
  
{'jsonrpc': '2.0', 'result': 'HTjMcuUHDoE3BkhcpWnLA6xWScDFzS7zQxWzUtffjMUkrZxaRjwrVjr8ta2Hr2uKxSUDXMzkLGiWbodgZk5DoEX', 'id': 190}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Tx on solscan  
  
The log from solscan:  
  
#1 Magic Eden NFT Marketplace instruction> Program Magic Eden NFT Marketplace consumed 8890 of 200000 compute units> Program returned error: Could not create program address with signer seeds: Provided seeds do not result in a valid address#2 System Program instruction#3 Token Program instruction  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Current problem: Added to the error in the log above the two new instructions (create account and set authority) are not inner instructions but just some instructions following the first instruction.  
  
I have used anchor-py to get the idl of the program, but it is not available (anchorpy.error.IdlNotFoundError: IDL not found for program: MEisE1HzehtrDpAAT8PnLHjpSSkRYakotTuJRPjTpo8).  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**C1:**Hello, I have the same exact problem, I cannot fetch the IDL because I get the error anchorpy.error.IdlNotFoundError how did you solve the issue?

2 **Answer**

**A1:**Disclaimer: I have never used the solana-py but looking at the accounts on the successful transaction I can see a difference. Account2 on the successful transaction is not an NFT token. ie. https://solscan.io/account/5KCdkEiCeQQ7XxZfFgQ2h4CaaMu72386LX7daxcSUyFo while on your failed transaction you are passing in https://solscan.io/token/3gS9AqTJ9adw23tZ87Hn1ccyYJ5KZ5tcoNQfYhCFu2e3which is your actual NFT.

**C1:**j0hn\_sm1th, you are right. Thank you. The account2 from my side is wrong. Digging deeper into the instructions i've found, that the account2 was generated on the fly via inner a inner instruction. I tried leaving the account2 value (and also all the following accounts) without success. I've also created the account myself and added it to the transaction without success. Is it possible to create the transaction in a way so that the new account is created as an inner instruction?

**C2:**Make sure when you actually create that account you are not just generating a keypair? You would have to call a create\_account transaction before the above 'transaction' or add a create account 'instruction' to the single 'transaction' so the account is actually initialised (unless it could be initialised internally on the metaplex program). Multiple instructions can be chained into a transaction. Also note: have you tried 'fetching the IDL' using the anchor CLI for the magic eden program? This would have required them to publish it, which I am not sure they would have done.

**C3:**I have added two instructions (see edited question) and tried to fetch the idl (without success, because it could not be found).

**C4:**Yep, the 'data' you are passing in must not be valid for more than 1 transaction. I assume it is some sort of seed which it will be trying to create PDA account for internally. As this is generated on the client \*ie. ME website) you could try reverse engineer the ME website js and find where/how the instruction data is occuring/created.

**A2:**thanks to this tweet I've found how to solve this issue:  
  
anchorpy.error.IdlNotFoundError: IDL not found for program  
  
After deploying your Anchor program, you have also to publish it's IDL using:  
  
anchor idl init <programId> -f <target/idl/program.json>  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]