Reading from Network

Now, let's explore how to read data from the Solana network. We'll fetch a few different accounts to understand the structure of a Solana account.

On Solana, all data is contained in what we call "accounts". You can think of data on Solana as a public database with a single "Accounts" table, where each entry in this table is an individual account.

Accounts on Solana can store "state" or "executable" programs, all of which can be thought of as entries in the same "Accounts" table. Each account has an "address" (public key) that serves as its unique ID used to locate its corresponding on-chain data.

Solana accounts contain either:

- State: This is data that's meant to be read from and persisted. It could be
- information about tokens, user data, or any other type of data defined within
- a program.
- Executable Programs: These are accounts that contain the actual code of Solana
- programs. They contain the instructions that can be executed on the network.

This separation of program code and program state is a key feature of Solana's Account Model. For more details, refer to theSolana Account Model page.

Fetch Playground Wallet#

Let's start by looking at a familiar account - your own Playground Wallet! We'll fetch this account and examine its structure to understand what a basic Solana account looks like.

Open Example 1#

Click thislink to open the example in Solana Playground. You'll see this code:

client.ts const address = pg.wallet.publicKey; const accountInfo = await pg.connection. getAccountInfo (address);

console. log (JSON . stringify (accountInfo, null , 2));

Explanation

Run Example 1#

In the Playground terminal, type therun command and hit enter:

Terminal run You should see details about your wallet account, including its balance in lamports, with output similar to the following:

Output

Explanation

Fetch Token Program#

Next, we'll examine the Token Extensions program, an executable program for interacting with tokens on Solana.

Open Example 2#

Click thislink to open the example in Solana Playground. You'll see this code:

client.ts import { PublicKey } from "@solana/web3.js" ;

const address = new PublicKey ("TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb"); const accountInfo = await pg.connection. getAccountInfo (address);

console. log (JSON . stringify (accountInfo, null , 2)); Instead of fetching your Playground wallet, here we fetch the address of the Token Extensions Program account.

Run Example 2#

Run the code using therun command in the terminal.

Terminal run Examine the output and how this program account differs from your wallet account.

Output

Explanation

Fetch Mint Account#

In this step, we'll examine a Mint account, which represents a unique token on the Solana network.

Open Example 3#

Click thislink to open the example in Solana Playground. You'll see this code:

client.ts import { PublicKey } from "@solana/web3.js" ;

const address = new PublicKey ("C33qt1dZGZSsqTrHdtLKXPZNoxs6U1ZBfyDkzmj6mXeR"); const accountInfo = await pg.connection. getAccountInfo (address);

console. log (JSON . stringify (accountInfo, null , 2)); In this example, we'll fetch the address of an existing Mint account on devnet.

Run Example 3#

Run the code using therun command.

Terminal run

Output

Explanation

Deserialize Mint Account Data#

To read thedata field from any account, you need to deserialize the data buffer into the expected data type. This is often done using helper functions from client libraries for a particular program.

Open this nextexample in Solana Playground. You'll see this code:

client.ts import { PublicKey } from "@solana/web3.js" ; import { getMint, TOKEN_2022_PROGRAM_ID } from "@solana/spl-token" ;

const address = new PublicKey ("C33qt1dZGZSsqTrHdtLKXPZNoxs6U1ZBfyDkzmj6mXeR"); const mintData = await getMint (pg.connection, address, "confirmed" , TOKEN_2022_PROGRAM_ID ,);

console. log (mintData); This example uses thegetMint helper function to automatically deserialize the data field of the Mint account.

Run the code using therun command.

Terminal run You should see the following deserialized Mint account data.

Output

Explanation

Previous «Quick Start Next Writing to Network»