Tokens on Solana

Tokens are digital assets that represent ownership over diverse categories of assets. Tokenization enables the digitalization of property rights, serving as a fundamental component for managing both fungible and non-fungible assets.

- Fungible Tokens represent interchangeable and divisible assets of the same
- type and value (ex. USDC).
- Non-fungible Tokens (NFT) represent ownership of indivisible assets (e.g.
- artwork).

This section will cover the basics of how tokens are represented on Solana. These are referred to as SPL (Solana Program Library) Tokens.

- The Token Program
- contains all the instruction logic for
- interacting with tokens on the network (both fungible and non-fungible).
- A Mint Account
- represents a specific type of token and stores
- global metadata about the token such as the total supply and mint authority
- (address authorized to create new units of a token).
- A Token Account
- keeps track of individual ownership of how
- many units of a specific type of token (mint account) are owned by a specific
- address.

Info There are currently two versions of the Token Program. The original Token Program and the Token Extensions Program (Token2022). The Token Extensions Program functions the same as the original Token Program, but with additional features and improvements. The Token Extensions Program is the recommended version to use for creating new tokens (mint accounts).

Key Points#

- Tokens represent ownership over either fungible (interchangeable) or
- non-fungible (unique) assets.
- The Token Program contains all instruction for interacting with both fungible
- and non-fungible tokens on the network.
- The Token Extensions Program is a new version of the Token Program that
- includes additional features while maintaining the same core functionalities.
- A Mint Account represents a unique token on the network and stores global
- metadata such as total supply.
- A Token Account tracks individual ownership of tokens for a specific mint
- account.
- An Associated Token Account is a Token Account created with an address derived
- from the owner's and mint account's addresses.

Token Program#

The Token Program contains all the instruction logic for interacting with tokens on the network (both fungible and non-fungible). All tokens on Solana are effectively data accounts owned by the Token Program.

You can find the full list of Token Program instructions here .

Token Program

A few commonly used instructions include:

- InitializeMint
- :
- Create a new mint account to represent a new type of token.
- InitializeAccount
- :
- Create a new token account to hold units of a specific type of token (mint).
- MintTo
- :
- Create new units of a specific type of token and add them to a token account.
- This increases the supply of the token and can only be done by the mint
- authority of the mint account.
- Transfer

- :
- Transfer units of a specific type of token from one token account to another.

Mint Account#

Tokens on Solana are uniquely identified by the address of aMint Account owned by the Token Program. This account is effectively a global counter for a specific token, and stores data such as:

- Supply: Total supply of the token
- Decimals: Decimal precision of the token
- Mint authority: The account authorized to create new units of the token, thus
- increasing the supply
- Freeze authority: The account authorized to freeze tokens from being
- transferred from "token accounts"

Mint Account

The full details stored on each Mint Account include the following:

pub struct Mint { /// Optional authority used to mint new tokens. The mint authority may only /// be provided during mint creation. If no mint authority is present /// then the mint has a fixed supply and no further tokens may be /// minted. pub mint_authority : COption < Pubkey

> , /// Total supply of tokens. pub supply : u64 , /// Number of base 10 digits to the right of the decimal place. pub decimals : u8 , /// Is true if this structure has been initialized pub is_initialized : bool , /// Optional authority to freeze token accounts. pub freeze_authority : COption < Pubkey , } For reference, here is a Solana Explorer link to theUSDC Mint Account .

Token Account#

To track the individual ownership of each unit of a specific token, another type of data account owned by the Token Program must be created. This account is referred to as aToken Account .

The most commonly referenced data stored on the Token Account include the following:

- Mint: The type of token the Token Account holds units of
- Owner: The account authorized to transfer tokens out of the Token Account
- Amount: Units of the token the Token Account currently holds

Token Account

The full details stored on each Token Account includes the following:

pub struct Account { /// The mint associated with this account pub mint : Pubkey , /// The owner of this account. pub owner : Pubkey , /// The amount of tokens this account holds. pub amount : u64 , /// If delegate is Some then delegated_amount represents /// the amount authorized by the delegate pub delegate : COption < Pubkey

> , /// The account's state pub state : AccountState , /// If is_native.is_some, this is a native token, and the value logs the /// rent-exempt reserve. An Account is required to be rent-exempt, so /// the value is used by the Processor to ensure that wrapped SOL /// accounts do not drop below this threshold. pub is_native : COption < u64 , /// The amount delegated pub delegated_amount : u64 , /// Optional authority to close the account. pub close_authority : COption < Pubkey , } For a wallet to own units of a certain token, it needs to create a token account for a specific type of token (mint) that designates the wallet as the owner of the token account. A wallet can create multiple token accounts for the same type of token, but each token account can only be owned by one wallet and hold units of one type of token.

Account Relationship

Info Note that each Token Account's data includes anowner field used to identify who has authority over that specific Token Account. This is separate from the program owner specified in theAccountInfo , which is the Token Program for all Token Accounts.

Associated Token Account#

To simplify the process of locating a token account's address for a specific mint and owner, we often use Associated Token Accounts.

An Associated Token Account is a token account whose address is deterministically derived using the owner's address and the mint account's address. You can think of the Associated Token Account as the "default" token account for a specific mint and owner.

It's important to understand that an Associated Token Account isn't a different type of token account. It's just a token account with a specific address.

Associated Token Account

This introduces a key concept in Solana development[Program Derived Address (PDA)](). Conceptually, a PDA provides a deterministic way to generate an address using some predefined inputs. This enables us to easily find the address of an account at a later time.

Here is a[Solana Playground]() example that derives the USDC Associated Token Account address and owner. It will always generate the[same address]() for the same mint and owner.

import { getAssociatedTokenAddressSync } from "@solana/spl-token" ;

const associatedTokenAccountAddress = getAssociatedTokenAddressSync (USDC_MINT_ADDRESS , OWNER_ADDRESS ,); Specifically, the address for an Associated Token Account is derived using the following inputs. Here is a[Solana Playground]() example that generates the same address as the previous example.

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

const [PDA , bump] = PublicKey. findProgramAddressSync ([OWNER_ADDRESS . toBuffer (), TOKEN_PROGRAM_ID . toBuffer (), USDC_MINT_ADDRESS . toBuffer (),], ASSOCIATED_TOKEN_PROGRAM_ID ,); For two wallets to hold units of the same type of token, each wallet needs its own token account for the specific mint account. The image below demonstrates what this account relationship looks like.

Accounts Relationship Expanded

Token Examples[#]()

The[spl-token CLI]() can be used to experiment with SPL tokens. In the examples below, we'll use the[Solana Playground]() terminal to run the CLI commands directly in the browser without having to install the CLI locally.

Creating tokens and accounts requires SOL for account rent deposits and transaction fees. If it is your first time using Solana Playground, created a Playground wallet and run thesolana airdrop command in the Playground terminal. You can also get devnet SOL using the public[web faucet]() .

solana airdrop 2 Runspl-token --help for a full description of available commands.

spl-token --help Alternatively, you can install the spl-token CLI locally using the following command. This requires first[installing Rust]() .

Info In the following sections, the account addresses displayed when you run the CLI command will differ from the example output shown below. Please use the address shown in your Playground terminal when following along. For example, the address output from thecreate-token is the mint account where your Playground wallet is set as the mint authority.

Create a New Token[#]()

To create a new token ([mint account]()) run the following command in the Solana Playground terminal.

spl-token create-token You should see an output similar to the following below. You can inspect both the token and transaction details on[Solana Explorer]() using theAddress andSignature .

In the example output below, the unique identifier (address) of the new token is99zqUzQGohamfYxyo8ykTEbi91iom3CLmwCA75FK5zTg .

Terminal Creating token 99zqUzQGohamfYxyo8ykTEbi91iom3CLmwCA75FK5zTg

Address: 99zqUzQGohamfYxyo8ykTEbi91iom3CLmwCA75FK5zTg Decimals: 9

Signature: 44fvKfT1ezBUwdzrCys3fvCdFxbLMnNvBstds76QZyE6cXag5NupBprSXwxPTzzjrC3cA6nvUZaLFTvmcKyzxrm1 New tokens initially have no supply. You can check the current supply of a token using the following command:

spl-token supply < TOKEN_ADDRES S

Running thesupply command for a newly created token will return a value of0 :

spl-token supply

99zqUzQGohamfYxyo8ykTEbi91iom3CLmwCA75FK5zTg Under the hood, creating a new Mint Account requires sending a transaction with two instructions. Here is a Javascript example on[Solana Playground]() .

1. Invoke the System Program to create a new account with enough space for the
2. Mint Account data and then transfer ownership to the Token Program.
3. Invoke the Token Program to initialize the data of the new account as a Mint
4. Account

Create Token Account#

To hold units of a particular token, you must first create atoken account . To create a new token account, use the following command:

spl-token create-account [OPTIONS] < TOKEN_ADDRESS

For example, running the following command in the Solana Playground terminal:

spl-token create-account

99zqUzQGohamfYxyo8ykTEbi91iom3CLmwCA75FK5zTg Returns the following output:

- AfB7uwBEsGtrrBqPTVqEgzWed5XdYfM1psPNLmf7EeX9
- is the address of the token
- account created to hold units of the token specified in thecreate-account
- command.

Terminal Creating account AfB7uwBEsGtrrBqPTVqEgzWed5XdYfM1psPNLmf7EeX9

Signature:
2BtrynuCLX9CNofFiaw6Yzbx6hit66pup9Sk7aFjwU2NEbFz7NCHD9w9sWhrCfEd73XveAGK1DxFpJoQZPXU9tS1 By default thecreate-account command creates anassociated token account with your wallet address as the token account owner.

You can create a token account with a different owner using the following command:

spl-token create-account --owner < OWNER_ADDRES S

< TOKEN_ADDRES S

For example, running the following command:

spl-token create-account --owner

2i3KvjDCZWxBsqcxBHpdEaZYQwQSYE6LXUMx5VjY5XrR 99zqUzQGohamfYxyo8ykTEbi91iom3CLmwCA75FK5zTg Returns the following output:

- Hmyk3FSw4cfsuAes7sanp2oxSkE9ivaH6pMzDzbacqmt
- is the address of the token
- account created to hold units of the token specified in thecreate-account
- command (99zqUzQGohamfYxyo8ykTEbi91iom3CLmwCA75FK5zTg
-) and owned by the
- address specified following the--owner
- flag
- (2i3KvjDCZWxBsqcxBHpdEaZYQwQSYE6LXUMx5VjY5XrR
-). This is useful when you need
- to create a token account for another user.

Terminal Creating account Hmyk3FSw4cfsuAes7sanp2oxSkE9ivaH6pMzDzbacqmt

Signature:
44vqKdfzspT592REDPY4goaRJH3uJ3Ce13G4BCuUHg35dVUbHuGTHvqn4ZjYF9BGe9QrjMfe9GmuLkQhSZCBQuEt Under the hood, creating an Associated Token Account requires a single instruction that invokes theAssociated Token Program . Here is a Javascript example onSolana Playground .

The Associated Token Program usesCross Program Invocations to handle:

- Invoking the System Program
- to create a new account using the provided PDA as the address of the new
- account
- Invoking the Token Program
- to initialize the Token Account data for the new account.

Alternatively, creating a new Token Account using a randomly generated keypair (not an Associated Token Account) requires sending a transaction with two instructions. Here is a Javascript example onSolana Playground .

1. Invoke the System Program to create a new account with enough space for the
2. Token Account data and then transfer ownership to the Token Program.
3. Invoke the Token Program to initialize the data of the new account as a Token
4. Account

Mint Tokens#

To create new units of a token, use the following command:

spl-token mint [OPTIONS] < TOKEN_ADDRESS

< TOKEN_AMOUNT

[--] [RECIPIENT_TOKEN_ACCOUNT_ADDRESS] For example, running the following command:

spl-token mint 99zqUzQGohamfYxyo8ykTEbi91iom3CLmwCA75FK5zTg 100 Returns the following output:

- 99zqUzQGohamfYxyo8ykTEbi91iom3CLmwCA75FK5zTg
- is the address of the mint
- account that tokens are being minted for (increasing total supply).
- AfB7uwBEsGtrrBqPTVqEgzWed5XdYfM1psPNLmf7EeX9
- is the address of your wallet's
- token account that units of the token are being minted to (increasing amount).

Terminal Minting 100 tokens Token: 99zqUzQGohamfYxyo8ykTEbi91iom3CLmwCA75FK5zTg Recipient: AfB7uwBEsGtrrBqPTVqEgzWed5XdYfM1psPNLmf7EeX9

Signature: 2NJ1m7qCraPSBAVxbr2ssmWZmBU9Jc8pDtJAnyZsZJRcaYCYMqq1oRY1gqA4ddQno3g3xcnny5fzr1dvsnFKMEqG To mint tokens to a different token account, specify the address of the intended recipient token account. For example, running the following command:

spl-token mint 99zqUzQGohamfYxyo8ykTEbi91iom3CLmwCA75FK5zTg 100 --

Hmyk3FSw4cfsuAes7sanp2oxSkE9ivaH6pMzDzbacqmt Returns the following output:

- 99zqUzQGohamfYxyo8ykTEbi91iom3CLmwCA75FK5zTg
- is the address of the mint
- account that tokens are being minted for (increasing total supply).
- Hmyk3FSw4cfsuAes7sanp2oxSkE9ivaH6pMzDzbacqmt
- is the address of the token
- account that units of the token are being minted to (increasing amount).

Terminal Minting 100 tokens Token: 99zqUzQGohamfYxyo8ykTEbi91iom3CLmwCA75FK5zTg Recipient: Hmyk3FSw4cfsuAes7sanp2oxSkE9ivaH6pMzDzbacqmt

Signature: 3SQvNM3o9DsTiLwcEkSPT1Edr14RgE2wC54TEjonEP2swyVCp2jPWYWdD6RwXUGpvDNUkKWzVBZVFShn5yntxVd7 Under the hood, creating new units of a token requires invoking theMintTo instruction on the Token Program. This instruction must be signed by the mint authority. The instruction mints new units of the token to a Token Account and increases the total supply on the Mint Account. Here is a Javascript example onSolana Playground .

Transfer Tokens#

To transfer units of a token between two token accounts, use the following command:

spl-token transfer [OPTIONS] < TOKEN_ADDRESS

< TOKEN_AMOUNT

< RECIPIENT_ADDRESS or RECIPIENT_TOKEN_ACCOUNT_ADDRES S

For example, running the following command:

spl-token transfer 99zqUzQGohamfYxyo8ykTEbi91iom3CLmwCA75FK5zTg 100 Hmyk3FSw4cfsuAes7sanp2oxSkE9ivaH6pMzDzbacqmt Returns the following output:

- AfB7uwBEsGtrrBqPTVqEgzWed5XdYfM1psPNLmf7EeX9
- is the address of the token
- account that tokens are being transferred from. This would be the address of
- your token account for the specified token being transferred.

- Hmyk3FSw4cfsuAes7sanp2oxSkE9ivaH6pMzDzbacqmt
- is the address of the token
- account that tokens are being transferred to.

Terminal Transfer 100 tokens Sender: AfB7uwBEsGtrrBqPTVqEgzWed5XdYfM1psPNLmf7EeX9 Recipient: Hmyk3FSw4cfsuAes7sanp2oxSkE9ivaH6pMzDzbacqmt

Signature: 5y6HVwV8V2hHGLTVmTmdySRiEUCZnWmkasAvJ7J6m7JR46obbGKCBqUFgLpZu5zQGwM4Xy6GZ4M5LKd1h6Padx3o Under the hood, transferring tokens requires invoking theTransfer instruction on the Token Program. This instruction must be signed by the owner of the sender's Token Account. The instruction transfers units of a token from one Token Account to another Token Account. Here is a Javascript example onSolana Playground .

It's important to understand that both the sender and recipient must have existing token accounts for the specific type of token being transferred. The sender can include additional instructions on the transaction to create the recipient's token account, which generally is the Associated Token Account.

Create Token Metadata#

The Token Extensions Program enables additional customizable metadata (such as name, symbol, link to image) to be stored directly on the Mint Account.

Info To use the Token Extensions CLI flags, ensure you have a local installation of the CLI, version 3.4.0 or later:

cargo install --version 3.4.0 spl-token-cli To create a new token with the metadata extension enabled, using the following command:

spl-token create-token --program-id TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb --enable-metadata The command returns the following output:

- BdhzpzhTD1MFqBiwNdrRy4jFo2FHFufw3n9e8sVjJczP
- is the address of the new token
- created with the metadata extension enabled.

Terminal Creating token BdhzpzhTD1MFqBiwNdrRy4jFo2FHFufw3n9e8sVjJczP under program TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb To initialize metadata inside the mint, please run spl-token initialize-metadata BdhzpzhTD1MFqBiwNdrRy4jFo2FHFufw3n9e8sVjJczP <YOUR_TOKEN_NAME> <YOUR_TOKEN_SYMBOL> <YOUR_TOKEN_URI>, and sign with the mint authority.

Address: BdhzpzhTD1MFqBiwNdrRy4jFo2FHFufw3n9e8sVjJczP Decimals: 9

Signature: 5iQofFeXdYhMi9uTzZghcq8stAaa6CY6saUwcdnELST13eNSifiuLbvR5DnRt311frkCTUh5oecj8YEvZSB3wfai Once a new token is created with the metadata extension enabled, use the following command to initialize the metadata.

spl-token initialize-metadata < TOKEN_MINT_ADDRES S

> < YOUR_TOKEN_NAM E
>
> < YOUR_TOKEN_SYMBOL
>
> < YOUR_TOKEN_URI
>
> The token URI is normally a link to off-chain metadata you want to associate with the token. You can find an example of the JSON formathere .

For example, running the following command will store the additional metadata directly on the specified mint account:

spl-token initialize-metadata

BdhzpzhTD1MFqBiwNdrRy4jFo2FHFufw3n9e8sVjJczP "TokenName" "TokenSymbol" "https://raw.githubusercontent.com/solana-developers/opos-asset/main/assets/DeveloperPortal/metadata.json" You can then look up the address of the mint account on an explorer to inspect the metadata. For example, here is a token created with the metadata extension enabled on theSolanaFm explorer.

You can learn more on theMetadata Extension Guide . For more details related to various Token Extensions, refer to the Token ExtensionsGetting Started Guide and theSPL documentation .

Previous «Cross Program Invocation Next Clusters & Endpoints»