

cmta.

# CMTAT Solana Specification

---

**Functional specifications of the CMTAT on Solana.**

CMTAT Solana version: v1.0.0  
First published: December 2025

# CMTAT Solana

---

This document contains the guidelines to deploy a CMTAT compliant token on [Solana](#).

## CMTAT Solana

Introduction

CMTAT

Solana

Solana Token introduction

Solana Token Program

Key Points

Features

Solana Extension (Token-2022)

Mint extensions

Account extensions

CMTAT - Solana Requirements

Comparison tab

CMTAT framework -> Solana token (Basic/Extension)

CMTAT extended -> Solana token (Basic/Extension)

Solana token basic -> CMTAT

Solana extensions -> CMTAT

Schema

Features

Base

Extensions

Access Control

Main difference with EVM Solidity version

Access control

ERC-20: `approve`

Burn / Transfer

Mint/Burn while pause

Enforcement (Freeze)

On-chain Metadata

CMTAT Deployment Guide (Token-2022)

Set up

Environment variable

Run a local blockchain

Generate Admin Keypair

Create Token with the required extensions

Example

Verification

Initialize On-Chain Metadata

Example

Create Token Account

Admin

User

Mint Initial Supply

Admin

User

Freeze / Unfreeze Accounts

Freeze

Transfer

- Unfreeze (thaw)
- Burn Tokens / Forced Transfer (Permanent Delegate)
  - Burn as admin
  - User as signer
  - Forced transfer
- Pause / Resume All Activity
  - Pause
  - Resume
- Update On-Chain Metadata
  - Remove a custom field
- Deactivate token
  - Burn all tokens
  - Deactivate Authorities
  - Close the Mint
- CMAT with whitelist
  - Command line
  - Example
    - Create token mint
    - Create token account
      - Instruction details
      - Account status
    - Thaw / unfreeze token account
      - New status
    - Update token mint
    - Create token account
  - Reference

## Introduction

---

### CMTAT

- The CMTA Standard Token for Securities (CMTAT) is an open standard for smart contracts designed specifically for the tokenization of financial instruments such as equity, debt and structured products and other transferable securities.
- CMTAT is blockchain agnostic in that it defines a set of functionalities that a security token should implement.

CMTAT specifications are available on CMTA website: [cmta.ch/standards/cmta-token-cmtat](https://cmta.ch/standards/cmta-token-cmtat)

### Solana

- Solana is a high-performance, Layer 1 blockchain optimized for finance and internet capital markets.
  - Solana enables parallel execution and deterministic finality natively on its base layer, ensuring high throughput without sacrificing user experience or composability.
  - The **Solana Virtual Machine (SVM)** is the runtime environment responsible for executing Solana programs—stateless smart contracts stored on-chain as executable accounts.
- Solana ships with a set of built-in programs that act as foundational components for most on-chain activity. These core programs fall into two categories:
- Native Programs, which provide low-level system and protocol functionality

- Solana Program Library (SPL) Programs, which offer higher-level, reusable standards such as token and associated account management.

Reference: [Solana - What is Solana](#), [Solana - tokenized equities](#) and [Solana - EVM vs. SVM: Smart Contracts](#)

## Solana Token introduction

---

Tokens on Solana are digital assets that represent ownership over diverse categories of assets. Tokenization enables the digitalization of property rights.

Tokens on Solana are referred to as SPL ([Solana Program Library](#)) Tokens.

## Solana Token Program

This program defines a common implementation for Fungible and Non Fungible tokens.

### Key Points

- [Token Programs](#) contain all instruction logic for interacting with tokens on the network (both fungible and non-fungible).
- A [Mint Account](#) represents a specific 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](#) tracks individual ownership of tokens for a specific mint account for a specific owner.
- An [Associated Token Account](#) is a Token Account created with an address derived from the owner and mint account addresses.

### Features

- Create a Token Mint – Initialize a new SPL Token mint.
- Create a Token Account – Open an account to hold SPL tokens.
- Mint Tokens – Generate new units of the token and assign them to an account.
- Transfer Tokens – Send tokens between accounts securely.
- Approve Delegate – Grant another account permission to manage tokens on behalf of the owner.
- Revoke Delegate – Remove delegate permissions from an account.
- Set Authority – Change the authority for a mint or token account (e.g., minting, freezing, account management).
- Burn Tokens – Permanently destroy tokens, reducing supply.
- Sync Native – Wrap native SOL into wrapped SOL (WSOL) for token program compatibility.
- Close Token Account – Close an SPL Token account and reclaim its SOL rent.
- Freeze Account – Halt activity on a token account.
- Thaw Account – Reactivate a previously frozen token account.

## Solana Extension (Token-2022)

The Token-2022 Program, also known as Token Extensions, is a superset of the functionality provided by the [Token Program](#).

Token extensions introduce a new set of ways to extend the normal token functionality.

- The original Token program brought the basic capabilities of minting, transferring and freezing tokens.

- The Token Extensions program includes the same features, but comes with additional features such as permanent delegate, custom transfer logic, extended metadata, and much more.
- The [Token Extensions program](#) has the programID `TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb` and is a superset of the original functionality provided by the [Token Program](#) at `TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA`.

The full list of available extensions on the Solana Token-2022 program is below:

## Mint extensions

Mint extensions currently include:

- transfer fees
- closing mint
- interest-bearing tokens
- non-transferable tokens
- permanent delegate
- transfer hook
- metadata pointer
- metadata
- group pointer
- group
- group member pointer
- group member
- scaled UI amount
- pausable
- confidential transfers
- confidential mint-burn

Confidential transfers and mint-burn will not be analyzed in this document. We could do that, however, in the near future.

## Account extensions

Account extensions currently include:

- memo required on incoming transfers
- immutable ownership
- default account state
- CPI guard

# CMTAT - Solana Requirements

---

## Comparison tab

This section contains four tables to indicate if a Solana token feature or an extension is required or optional to align with the CMTAT framework's required or optional features.

## CMTAT framework -> Solana token (Basic/Extension)

In the below table, the CMTAT framework required features are mapped to Solana token features.

| CMTAT framework mandatory functionalities | Solana token basic features             | Solana Extension   | CMTAT Solidity corresponding features   |
|---|---|--|---|
| Know total supply                         | Query the <a href="#">Mint Account</a>  | ☒  | ERC20 <code>totalSupply</code>  |
| Know balance                              | Query the <a href="#">Token Account</a> | ☒  | ERC20 <code>balanceOf</code>  |
| Transfer tokens                           | <a href="#">Transfer tokens</a>         | ☒  | ERC20 <code>transfer</code>   |
| Create tokens (mint)                      | <a href="#">Mint tokens</a>             | ☒  | Mint/batchMint  |
| Cancel tokens (force burn)                | ☒                                       | <a href="#">Permanent Delegate</a>   | burn/batchBurn  |
| Pause tokens                              | ☒                                       | <a href="#">Pausable</a> ( <i>pause</i> )<br><i>(Nb. During this time, it is normally not possible to transfer, mint or burn tokens.)</i><br><i>To enable mint and burn during pause a transfer hook must be used. See here.</i> | Pause<br><i>(Nb. With CMTAT Solidity it is still possible to burn and mint while transfers are paused.)</i> |
| Unpause tokens                            | ☒                                       | <a href="#">Pausable</a> ( <i>resume</i> )   | <code>unpause</code>  |
| Deactivate contract                       | ☒                                       | <a href="#">Mint Close Authority</a><br>(Burn all tokens, close Mint Account, eventually revoke authorities)   | <code>deactivateContract</code>   |
| Freeze                                    | <a href="#">Freeze Account</a>          | ☒  | <code>setAddressFrozen</code><br>(previously <code>freeze</code> )  |
| Unfreeze                                  | <a href="#">Thaw Account</a>            | ☒  | <code>setAddressFrozen</code><br>(previously <code>unfreeze</code> )  |
| Name attribute                            | ☒                                       | <a href="#">Metadata</a> , <a href="#">Metadata Pointer</a>  | ERC20 <code>name</code> attribute   |

| CMTAT framework mandatory functionalities   | Solana token basic features | Solana Extension  | CMTAT Solidity corresponding features |
|---|-----------------------------|---|---------------------------------------|
| Ticker symbol attribute                     | ☒                           | <a href="#">Metadata</a> , <a href="#">Metadata Pointer</a> | ERC20 <code>symbol</code> attribute   |
| Token ID attribute                          | ☒                           | <a href="#">Metadata</a> , <a href="#">Metadata Pointer</a> | <code>tokenId</code>                  |
| Reference to legally required documentation | ☒                           | <a href="#">Metadata</a> , <a href="#">Metadata Pointer</a> | <code>terms</code>                    |

## CMTAT extended -> Solana token (Basic/Extension)

In the below table, the CMTAT framework optional features are mapped to Solana token features.

| CMTAT Functionalities      | Solana token basic features  | Solana Extension                      | CMTAT Solidity corresponding features                                 |
|----------------------------|--|---------------------------------------|---|
| force Transfer             | ☒  | <a href="#">Permanent Delegate</a>    | <code>forcedTransfer</code>   |
| freeze partial token       | ☒  | ☒                                     | <code>freezePartialTokens</code> / <code>unfreezePartialTokens</code> |
| Whitelisting               | ☒  | <a href="#">Default Account State</a> | CMTAT Allowlist / CMTAT with rule whitelist                           |
| RuleEngine / transfer hook | ☒  | <a href="#">Transfer Hook</a>         | CMTAT with RuleEngine   |
| On-chain snapshot          | ☒  | ☒                                     | SnapshotEngine or dedicated deployment version                        |
| Upgradability              | N/A  | N/A                                   | CMTAT Upgradeable version   |
| Feepayer/gasless           | N/A<br>Fee sponsorship is a native feature in Solana. See <a href="#">Solana - Fee Sponsorship</a> | N/A                                   | CMTAT with ERC-2771 module  |

## Solana token basic -> CMTAT

In the below table, the Solana token basic features are mapped to the CMTAT framework.

| Solana Token features | Description | Solana CMTAT Required | CMTAT Required | CMTAT optional | CMTAT corresponding features | Note |
|-----------------------|-------------|-----------------------|----------------|----------------|------------------------------|------|
|-----------------------|-------------|-----------------------|----------------|----------------|------------------------------|------|

| Solana Token features                  | Description   | Solana CMTAT Required               | CMTAT Required                      | CMTAT optional                      | CMTAT corresponding features       | Note  |
|--|---|-------------------------------------|-------------------------------------|-------------------------------------|------------------------------------|---|
| <a href="#">Create a Token Mint</a>    | A mint account uniquely represents a token on Solana and stores its global metadata.  | <input checked="" type="checkbox"/> | N/A                                 | N/A                                 | N/A                                | Correspond to deploy the smart contract on EVM chain  |
| <a href="#">Create a Token Account</a> | A token account stores your balance of a specific token.  | <input checked="" type="checkbox"/> | N/A                                 | N/A                                 | N/A                                | No relevant for EVM based blockchain  |
| <a href="#">Mint Tokens</a>            | Minting creates new units of a token using the <code>MintTo</code> instruction.   | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | mint/batchMint                     |   |
| <a href="#">Transfer Tokens</a>        | Token transfers move tokens between token accounts of the same mint.  | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | ERC20 transfer                     |   |
| <a href="#">Approve Delegate</a>       | The <code>ApproveChecked</code> instruction grants another account (the delegate) permission to transfer a specific amount of tokens from your token account. | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | ERC-20 approve                     | Contrary to Ethereum (EVM), this is often not needed in Solana since you can do actual DVP in one transaction due to native instruction batching. See <a href="#">solana.com - transactions</a> |
| <a href="#">Revoke Delegate</a>        | The <code>Revoke</code> instruction removes all transfer permissions from the currently approved delegate.  | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | ERC-20 approve                     |   |
| <a href="#">Set Authority</a>          | The <code>SetAuthority</code> instruction changes or revokes authorities on mints and token accounts.   | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | RBAC system <code>grantRole</code> |   |
| <a href="#">Burn Tokens</a>            | The <code>BurnChecked</code> instruction permanently destroys tokens by reducing the balance in a token account.  | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <code>burnFrom</code>              |   |

| Solana Token features               | Description   | Solana CMTAT Required               | CMTAT Required                      | CMTAT optional                      | CMTAT corresponding features                  | Note   |
|-------------------------------------|---|-------------------------------------|-------------------------------------|-------------------------------------|---|--|
| <a href="#">Sync Native</a>         | The <code>SyncNative</code> instruction synchronizes a wrapped SOL (WSOL) token account balance with the actual SOL (lamports) stored in it.            | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/>           |  |
| <a href="#">Close Token Account</a> | The <code>CloseAccount</code> instruction permanently closes a token account and transfers all remaining SOL (rent) to a specified destination account. | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/>           | Solana specific features (but relevant to implement) |
| <a href="#">Freeze Account</a>      | The <code>FreezeAccount</code> instruction prevents all token transfers or token burns from a specific token account.                                   | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <code>setAddressFrozen(prev. freeze)</code>   |  |
| <a href="#">Thaw Account</a>        | The <code>ThawAccount</code> instruction reverses a freeze, restoring full functionality to a previously frozen token account.                          | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <code>setAddressFrozen(prev. unfreeze)</code> |  |

## Solana extensions -> CMTAT

In the below table, the Solana extension features are mapped to the CMTAT framework.

| Solana Extension                        | Description  | CMTAT Required (Core features)      | CMTAT optional                      | CMTAT Solidity corresponding features       |
|---|--|-------------------------------------|-------------------------------------|---|
| <a href="#">Mint Close Authority</a>    | Close mint account   | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <code>deactivateContract</code>             |
| <a href="#">Transfer Fees</a>           | Transferring tokens with a transfer fee                                  | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/>         |
| <a href="#">Default Account State</a>   | Force all new token accounts to be frozen.                               | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | CMTAT Allowlist / CMTAT with rule whitelist |
| <a href="#">Immutable Owner</a>         | Impossible to reassign ownership of an account.                          | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/>         |
| <a href="#">Non-Transferable Tokens</a> | Allows for "soul-bound" tokens that cannot be moved to any other entity. | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/>         |

| Solana Extension                          | Description   | CMTAT Required (Core features)      | CMTAT optional                      | CMTAT Solidity corresponding features            |
|---|---|-------------------------------------|-------------------------------------|--|
| <a href="#">Required Memo on Transfer</a> | Enforces that all incoming transfers must have an accompanying memo instruction right before the transfer instruction.  | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | CMTAT ERC-1363                                   |
| <a href="#">Interest-Bearing Tokens</a>   | Using the <code>InterestBearingMint</code> extension and the <code>amount_to_ui_amount</code> instruction, you can set an interest rate on your token and fetch its amount with interest at any time. | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/>              |
| <a href="#">Permanent Delegate</a>        | Allows to specify a permanent account delegate for a mint. This authority can burn or transfer any amount of tokens.  | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Forced transfer                                  |
| <a href="#">CPI Guard</a>                 | CPI Guard is an extension that prohibits certain actions inside cross-program invocations.  | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | N/A<br>(Solana specific extension)               |
| <a href="#">Transfer Hook</a>             | The Transfer Hook Interface is designed to allow token creators to "hook" additional functionality into token transfers.  | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | CMTAT with RuleEngine                            |
| <a href="#">Metadata Pointer</a>          | Allows a token creator to designate an address that describes the canonical metadata.   | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | N/A  |
| <a href="#">Metadata</a>                  | Allows a mint creator to include their token's metadata directly in the mint account.   | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | ERC20 name & symbol terms attributes (uri, hash) |
| <a href="#">Group Pointer</a>             | Allows a token creator to designate a group account that describes the mint.  | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/>              |

| Solana Extension        | Description  | CMTAT Required (Core features)      | CMTAT optional                      | CMTAT Solidity corresponding features  |
|-------------------------|--|-------------------------------------|-------------------------------------|--|
| <u>Group</u>            | Token-2022 supports grouping of tokens through the group extension.  | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/>  |
| <u>Member Pointer</u>   | The member pointer allows a token creator to designate a member account that describes the mint.                   | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/>  |
| <u>Member</u>           | The configurations for a member (group address and the member's number) can be stored directly in the mint itself. | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/>  |
| <u>Pausable</u>         | "Pause" all activity. During this time, it is not possible transfer, mint, or burn tokens.                         | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <p><code>pause</code><br/>With CMTAT Solidity, it is possible to burn and mint while transfers are paused.</p> |
| <u>Scaled UI Amount</u> | Change how the UI amount of tokens are represented   | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/>  |

## Schema

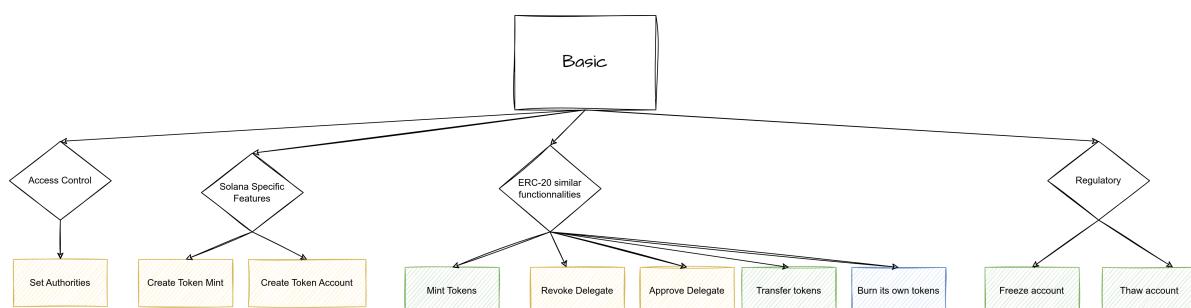
### Features

Not directly mentioned in the standard

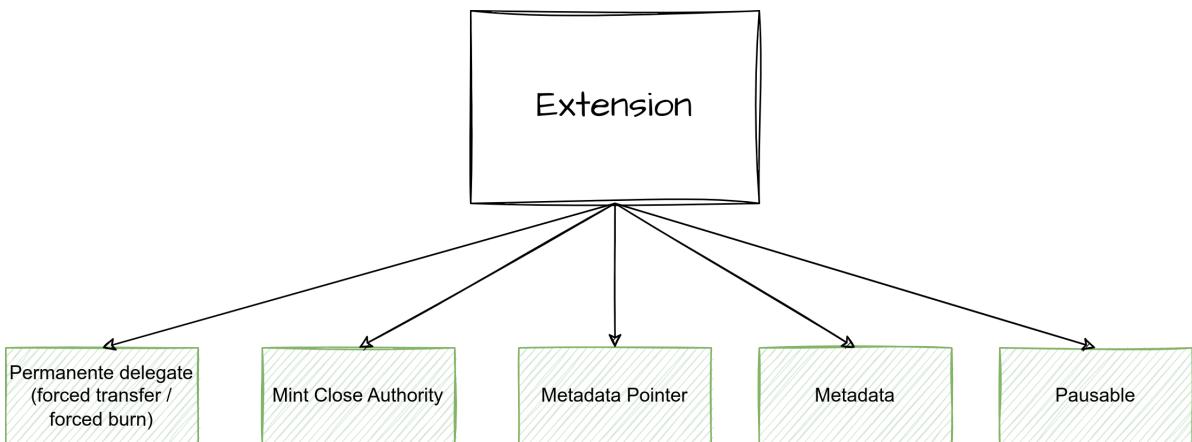
Required by CMTA specification

Optional features

### Base

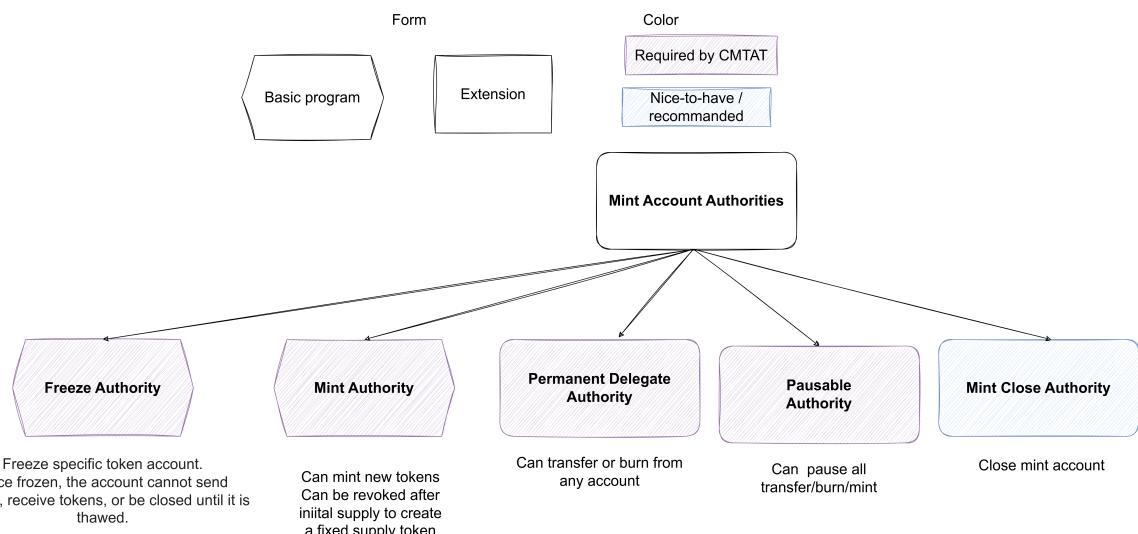


## Extensions



## Access Control

Here is a schema representing the different authorities:



## Main difference with EVM Solidity version

This section compares the features and behavior of the CMTAT specification on Solana (the present document) with the CMTAT implementation for Ethereum and other EVM-compatible blockchains written in Solidity. The latter is also available on GitHub: [CMTA/CMTAT](#).

The Ethereum Virtual Machine (EVM) is the decentralized computation environment that executes smart-contract code on Ethereum and other EVM-compatible blockchains. More information can be found on [Ethereum.org](#).

Here is a summary tab of the main difference:

| <b>ID</b> | <b>Functionalities</b>            | <b>CMTAT Solidity</b>               | <b>CMTAT Solana specification</b>   |
|-----------|-----------------------------------|-------------------------------------|-------------------------------------|
| 1         | Mint while pause                  | <input checked="" type="checkbox"/> | <input type="checkbox"/>            |
| 2         | Burn while pause                  | <input checked="" type="checkbox"/> | <input type="checkbox"/>            |
| 3         | Self Burn                         | <input type="checkbox"/>            | <input checked="" type="checkbox"/> |
| 4         | Standard burn on a frozen address | <input type="checkbox"/>            | <input checked="" type="checkbox"/> |

| <b>Id</b> | <b>Functionalities</b>                       | <b>CMTAT Solidity</b>               | <b>CMTAT Solana specification</b>   |
|-----------|--|-------------------------------------|-------------------------------------|
| 5         | Forced burn tokens with a dedicated function | <input checked="" type="checkbox"/> | <input type="checkbox"/>            |
| 6         | Flexible metadata architecture               | <input type="checkbox"/>            | <input checked="" type="checkbox"/> |

**Note:** 1,2,3 and 4 can be also achieved on Solana through custom [transfer hooks](#).

## Access control

- With the Solidity version, you can have a super admin which delegates several tasks to different addresses through different roles, for example the minter role to mint tokens.
- On Solana, you can also delegate roles, but it is not possible to delegate them while keeping a super admin.

Note: Similar to forced transfer above, if the role authorities are themselves smart contracts you can create complex control structures with multisigs, admin accounts etc. In order to have a super admin, you can designate all roles to one smart contract which then has its own rules as to who has authority over which role.

## ERC-20: `approve`

As on the EVM, Solana tokens also support delegated spending authority through the `Approve` and `Revoke Delegate` instructions.

However, unlike the EVM, this mechanism is often unnecessary on Solana because native instruction batching enables true Delivery-versus-Payment (DvP) to occur in a *single transaction*.

This is a significant usability and security advantage: the EVM's approval model is frequently seen as unfriendly to users and exposes them to risks, especially when decentralized applications request broad or unlimited approvals through their front-ends.

See [solana.com - transactions](#)

## Burn / Transfer

- On Solana, there is no difference between a regular transfer and a forced transfer, while on EVM it is two distinct functions. Same applies to a force burn.
  - Nevertheless, the signer of the transaction will be different if it is a forced transfer (delegate authority) or a regular transfer (token holder signer).
- Forced transfer is an optional feature of CMTAT. While the Solidity version allows to only include the force burn, on Solana, it is not possible to include the force burn without the force transfer.
  - It is possible to separate force burn from force transfer if the permanent delegate is itself a smart contract that would only allow for one of the two options.
- Contrary to the Solidity version, token holder can burn by default their own tokens.

## Mint/Burn while pause

- With Solidity, you can still burn and mint tokens while pausing regular transfers.
- This is not the case on Solana where the pause will also apply to mint and burn operations in addition to regular transfers.

Note: In order to enable this, you can use a [transfer hook](#) that is set to a program that just fails every transfer. This way you retain all other administrative features but prevent anyone from transferring tokens.

## Enforcement (Freeze)

- With Solidity, it is not possible to burn tokens on a frozen address through regular burn. The issuer must use `forcedBurn` or `forcedTransfer` to do it.
- On Solana, it is also not possible to burn tokens held in a frozen account. However, unlike in Solidity, the issuer cannot perform a forced transfer while the account remains frozen—even when acting through a delegate authority. Instead, Solana's ability to batch multiple instructions in a single transaction provides a practical workaround: the issuer can thaw (unfreeze) the account and then transfer the tokens via the delegate authority within the same transaction.

## On-chain Metadata

- In Solidity, each on-chain metadata attribute—such as `tokenId` or `terms`—must be explicitly defined in the smart contract before deployment, including the functions used to set and retrieve those values. Once deployed, the structure of these attributes is fixed.
- On Solana, metadata management is more flexible. Attributes can be added or defined after deployment using the Solana Token Extensions [Metadata](#) feature, allowing issuers to evolve or enrich token metadata without modifying the original program.

## CMTAT Deployment Guide (Token-2022)

---

This guide includes the deployment of a token on Solana which respects the CMTAT specification and contains the following features

- Mint tokens
- Freeze/unfreeze accounts
- Burn and forced transfer (Permanent Delegate)
- Pause/resume token activity
- Token information
  - Token name and Symbol
  - On-chain metadata with custom fields (`termsHash`, jurisdiction, issuer)
- Deactivate the token
  - close mint

All the operations have been made on a Linux machine through the SPL command line on a local Solana blockchain

# Set up

## Environment variable

To simplify command, different values are stored in the current bash session environment.

| Variable             | Description  | Value  |
|----------------------|--|--|
| ADMIN_SOLANA_KEYPAIR | Path to the JSON file containing the admin keypair   | Public Address<br>Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj |
| USER_SOLANA_KEYPAIR  | Path to the JSON file containing the user keypair    | Public address<br>9Grr8jKaZUji1VGj3uBBE3vWBmRbf48CGUchAUfxrqk  |
| TOKEN_MINT           | Token address  | JA6iL96LYav6GaMS1NnrZRTYA98PjfYQyu9DBwxdb2b2                   |
| ADMIN_TOKEN_ACCOUNT  | Admin Token account for the corresponding TOKEN_MINT | ApXcDVWCXhPgfaMUYqQ85JcYmyxqecqDPKXPz8B2fjqK                   |
| USER_TOKEN_ACCOUNT   | User Token account for the corresponding TOKEN_MINT  | diza8NKEJ7wTf31meXA9X2pyQnBsTnV4jsYUmwfFAK1                    |

## Run a local blockchain

```
solana-test-validator
```

```
Notice! No wallet available. `solana airdrop` localnet SOL after creating one

Ledger location: test-ledger
Log: test-ledger/validator.log
: Initializing...
Waiting for fees to stabilize 1...
: Initializing...
Waiting for fees to stabilize 2...
Identity: BwarQj5LnJu2GAySNPv26rn2QPptKUebgdNEQi4znxdt
Genesis Hash: HuB1STSJvb9Q7gLH9camtyoJTfNwtxndz5Kn2JBsuy7h
Version: 2.3.11
Shred Version: 50459
Gossip Address: 127.0.0.1:8000
TPU Address: 127.0.0.1:8003
JSON RPC URL: http://127.0.0.1:8899
WebSocket PubSub URL: ws://127.0.0.1:8900
```

## Generate Admin Keypair

This keypair will act as the mint authority, freeze authority, pause authority, and update authority for your token.

**Important:** The following command generates a new keypair for demonstration/example purposes.

Do not use this keypair in production, as it is stored locally and not secured in a hardware wallet or secure key management system. For production, use a **secure key management solution** (e.g., hardware wallet, secure enclave, or custodian service).

```

# Generate a new Solana keypair for admin authority
solana-keygen new -o test_admin.json
# Export environment variable to use this keypair with SPL commands export
export ADMIN_SOLANA_KEYPAIR=test_admin.json
# Generate a new Solana keypair for a test user
solana-keygen new -o test_user.json
# Export environment variable to use this keypair with SPL commands
export USER_SOLANA_KEYPAIR=test_user.json

# Change the signer in the config for the admin solana config
solana config set -k $ADMIN_SOLANA_KEYPAIR
# Check configuration
solana config get
# Check default address
solana address --keypair $ADMIN_SOLANA_KEYPAIR
solana address

#Change url to local blockchain/validator
solana config set --url http://127.0.0.1:8899

#Aidrop SOL token to pay gas fees locally
solana airdrop 1000 $USER_SOLANA_KEYPAIR
solana airdrop 1000 $ADMIN_SOLANA_KEYPAIR

```

- `test_admin.json` is the local file storing the token admin keypair.
- `test_user.json` is the local file storing the test user (token holder) keypair.
- `SOLANA_KEYPAIR` environment variable allows CLI tools (like `spl-token`) to reference it automatically.

## Create Token with the required extensions

```

spl-token create-token \
  --program-id
TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb \
  --decimals <decimals> \
  --mint-authority < Solana Keypair> \
  --enable-permanent-delegate \
  --enable-pause \
  --enable-close \
  --enable-metadata \
  --enable-freeze

```

### Note

- `TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb` is the programId of the Token Extensions program. See [Getting Started with Token Extensions](#)
- `decimals`: as part of CMTA specification, decimal numbers must be set to zero (which means that the tokens admit no fractional parts), unless the law governing the tokenized security allows the transfer of fractions.
- You can decide to use a different keypair address for the `mint-authority`
  - At deployment, the command-line does not allow you to set a different key for the other authorities (`freeze`, `pause`, `delegate`)

## Example

### Command

```
spl-token create-token --program-id  
TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb --decimals 0 --mint-authority  
$ADMIN_SOLANA_KEYPAIR --enable-permanent-delegate --enable-pause --  
enable-close --enable-metadata --enable-freeze
```

### Result

Creating token JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2 under program TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb

To initialize metadata inside the mint, please run `spl-token initialize-metadata`

`JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2 <YOUR_TOKEN_NAME>`

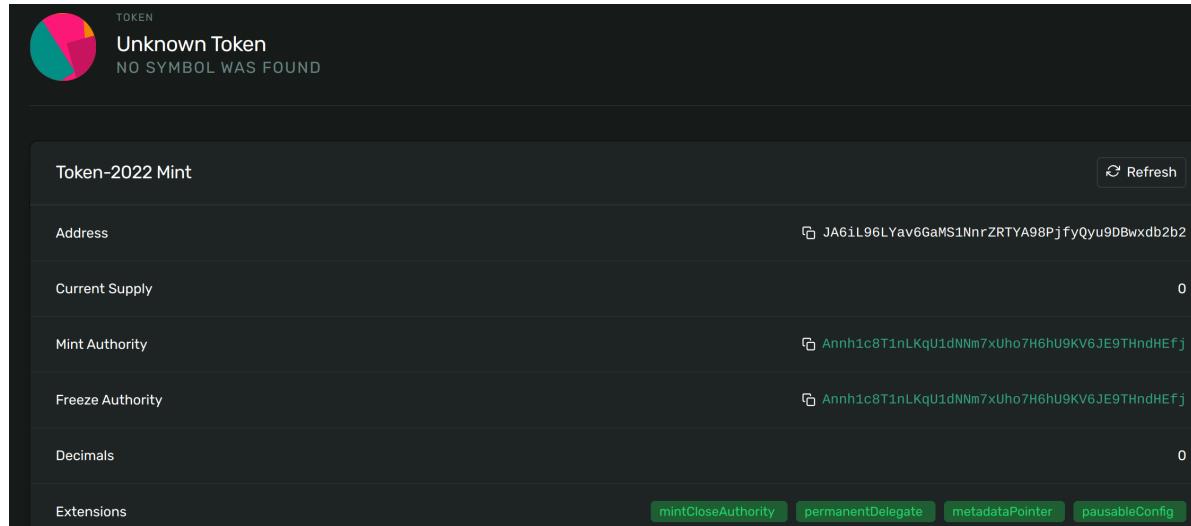
`<YOUR_TOKEN_SYMBOL> <YOUR_TOKEN_URI>`, and sign with the mint authority.

Address: JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2

Decimals: 0

Save the mint address as `TOKEN_MINT`.

```
export TOKEN_MINT=JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2
```



### Verification

```
spl-token display [OPTIONS] <TOKEN_ADDRESS>
```

```
spl-token display $TOKEN_MINT
```

### Result

## SPL Token Mint

Address: JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2

Program: TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb

Supply: 0

Decimals: 0

Mint authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Freeze authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

### Extensions

Close authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Permanent delegate: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

### Metadata Pointer:

Authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Metadata address: JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2

## Initialize On-Chain Metadata

```
spl-token initialize-metadata --url <url> \ --program-2022 --mint-authority
<Solana Keypair> \ --update-authority <Solana Keypair> <name> <symbol> <Token
URI>
```

### Note

- The CMTA specification only includes the token name, symbol and the terms uri (if relevant)
- Terms uri: Reference to any legally required documentation about the distributed ledger or the smart contract, such as the tokenization terms, the terms of the instrument and other relevant documents (e.g. prospectus or key information document).
- Token-2022 Metadata Structure. See [Token metadata](#)

| Field                         | Type                               | Description   | Example                      |
|-------------------------------|------------------------------------|---|------------------------------|
| <code>update_authority</code> | <code>OptionalNonZeroPubkey</code> | Public key allowed to update metadata. If unset, metadata is immutable. | <code>7gH...kP1</code>       |
| <code>mint</code>             | <code>Pubkey</code>                | The mint address this metadata belongs to (prevents spoofing).          | <code>&lt;address&gt;</code> |
| <code>name</code>             | <code>String</code>                | Human-readable name of the token.                                       | <code>CMTAT Token</code>     |
| <code>symbol</code>           | <code>String</code>                | Short token symbol ( $\leq 10$ chars).                                  | <code>CMTAT</code>           |

| Field                            | Type                                     | Description   | Example                                 |
|----------------------------------|--|---|---|
| <code>uri</code>                 | <code>String</code>                      | URI pointing to extended metadata or compliance docs (e.g. JSON file, website).           | <code>https://cmta.ch/token.json</code> |
| <code>additional_metadata</code> | <code>Vec&lt;(String, String)&gt;</code> | Flexible key-value pairs for extra fields such as compliance info, issuer, or terms hash. | <code>("termsHash", "Qm123...")</code>  |

## Example

```
spl-token initialize-metadata --program-2022 --mint-authority
$ADMIN_SOLANA_KEYPAIR --update-authority $ADMIN_SOLANA_KEYPAIR $TOKEN_MINT
CMTATTOKEN CMTATTOKEN https://cmta.ch/token.json
```

The screenshot shows the Token-2022 Mint interface with the following details:

- Address:** JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwdx2b2b2
- Current Supply:** 0
- Mint Authority:** Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj
- Freeze Authority:** Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj
- Decimals:** 0
- Extensions:** mintCloseAuthority, permanentDelegate, metadataPointer, pausableConfig, tokenMetadata

The screenshot shows the Token-2022 Program: Initialize Token Metadata interface with the following details:

- Metadata:** JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwdx2b2b2
- Mint:** JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwdx2b2b2
- MintAuthority:** Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj
- Name:** CMTATTOKEN
- Symbol:** CMTATT
- UpdateAuthority:** Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj
- Uri:** https://cmta.ch/token.json

## Verification

```
spl-token display $TOKEN_MINT
```

SPL Token Mint

Address: JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2

Program: TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb

Supply: 0

Decimals: 0

Mint authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Freeze authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Extensions

Close authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Permanent delegate: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Metadata Pointer:

Authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Metadata address: JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2

Metadata:

Update Authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Mint: JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2

Name: CMTATTOKEN

Symbol: CMTAT

URI: <https://cmta.ch/token.json>

## Create Token Account

```
spl-token create-account [OPTIONS] <TOKEN_MINT_ADDRESS> [ACCOUNT_KEYPAIR]
```

### Options

--program-2022

Use token extension program token 2022 with program id:

TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb

--owner <OWNER\_ADDRESS>

Address of the primary authority controlling a mint or account. Defaults to the client keypair address.

## Admin

Create the token account for the admin

```
spl-token create-account --program-2022 $TOKEN_MINT --owner
$ADMIN_SOLANA_KEYPAIR
```

| Token-2022 Account |  |
|--------------------|--|
| Address            | ApXcDVwCXhPgFAmUYqQ85JcYmyxqecqDPKXPz8B2fjqK |
| Mint               | JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2 |
| Owner              | Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj |
| Token balance      | 0  |
| Status             | Initialized                                  |
| Extensions         | ImmutableOwner pausableAccount               |

## Result

Creating account ApXcDVWCXhPgfAmUYqQ85JcYmyxqecqDPKXPz8B2fjqK

We export the token account created in an environment variable:

```
export ADMIN_TOKEN_ACCOUNT=ApXcDVWCXhPgfAmUYqQ85JcYmyxqecqDPKXPz8B2fjqK
```

## User

Create a user token account

```
# Change default key to USER
solana config set -k $USER_SOLANA_KEYPAIR
# Create account
spl-token create-account --program-2022 $TOKEN_MINT --owner
$USER_SOLANA_KEYPAIR
#rollback change
solana config set -k $ADMIN_SOLANA_KEYPAIR
```

Result:

Creating account diZa8NKEJ7wTf31meXA9X2pyQnBsTnV4jsYUmwfFAK1

We export the token account created in an environment variable:

```
export USER_TOKEN_ACCOUNT=diZa8NKEJ7wTf31meXA9X2pyQnBsTnV4jsYUmwfFAK1
```

| Token-2022 Account |  |
|--------------------|--|
| Address            | diZa8NKEJ7wTf31meXA9X2pyQnBsTnV4jsYUmwfFAK1  |
| Mint               | JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2 |
| Owner              | 9Grr8jKaZUji1VGj3uBBE3vWBmRbf48CGUchAUfxrqk  |
| Token balance      | 0  |
| Status             | Initialized                                  |
| Extensions         | immutableOwner pausableAccount               |

## Mint Initial Supply

```
spl-token mint [OPTIONS] <TOKEN_MINT_ADDRESS> <TOKEN_AMOUNT> [--]
[RECIPIENT_TOKEN_ACCOUNT_ADDRESS]
```

Example

## Admin

```
spl-token mint --program-2022 --mint-authority $ADMIN_SOLANA_KEYPAIR  
$TOKEN_MINT 1000 $ADMIN_TOKEN_ACCOUNT
```

## Result

Minting 1000 tokens

Token: JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2

Recipient: ApXcDVWCXhPgfAmUYqQ85JcYmyxqecqDPKXPz8B2fjqK

| Token-2022 Account |  |
|--------------------|--|
| Address            | ApXcDVWCXhPgfAmUYqQ85JcYmyxqecqDPKXPz8B2fjqK |
| Mint               | JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2 |
| Owner              | Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj |
| Token balance      | 1000   |
| Status             | Initialized                                  |
| Extensions         | immutableOwner pausableAccount               |

## Verification

- Check the supply

```
spl-token supply $TOKEN_MINT
```

Result: 1000

| Token-2022 Mint  |   |
|------------------|---|
| Address          | JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2                                      |
| Current Supply   | 1,000   |
| Mint Authority   | Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj                                      |
| Freeze Authority | Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj                                      |
| Decimals         | 0   |
| Extensions       | mintCloseAuthority permanentDelegate metadataPointer pausableConfig tokenMetadata |

- Check the token account balance

```
spl-token balance [OPTIONS] [TOKEN_MINT_ADDRESS]
```

```
spl-token balance --program-2022 --address $ADMIN_TOKEN_ACCOUNT
```

Result: 1000

## User

```
spl-token mint --program-2022 --mint-authority $ADMIN_SOLANA_KEYPAIR  
$TOKEN_MINT 300 $USER_TOKEN_ACCOUNT
```

## Result

Minting 300 tokens

Token: JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2

Recipient: diZa8NKEJ7wTf31meXA9X2pyQnBsTnV4jsYUmwfFAK1

```
spl-token balance --program-2022 --address $USER_TOKEN_ACCOUNT
```

Result: 300

```
spl-token supply $TOKEN_MINT
```

Result: 1300

| Token-2022 Account |  |
|--------------------|--|
| Address            | diZa8NKEJ7wTf31meXA9X2pyQnBsTnV4jsYUmwfFAK1  |
| Mint               | JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2 |
| Owner              | 9Grr8jKaZUji1VGj3uBBE3vwBmRbf48CGUchAUfxrqk  |
| Token balance      | 300  |
| Status             | Initialized                                  |
| Extensions         | immutableOwner pausableAccount               |

## Freeze / Unfreeze Accounts

The Mint may also contain a `freeze_authority` which can be used to issue `FreezeAccount` instructions that will render an Account unusable.

- Token instructions that include a frozen account will fail until the Account is thawed using the `ThawAccount` instruction.
- The `SetAuthority` instruction can be used to change a Mint's `freeze_authority`.
- If a Mint's `freeze_authority` is set to `None` then account freezing and thawing is permanently disabled and all currently frozen accounts will also stay frozen permanently.

```
spl-token freeze [OPTIONS] <TOKEN_ACCOUNT_ADDRESS>  
spl-token thaw [OPTIONS] <TOKEN_ACCOUNT_ADDRESS>
```

## Freeze

```
spl-token freeze --freeze-authority $ADMIN_SOLANA_KEYPAIR $USER_TOKEN_ACCOUNT
```

## Result

Freezing account: diZa8NKEj7wTf31meXA9X2pyQnBsTnV4jsYUmwfFAK1

Token: JA6iL96LYav6GaMS1NnrZRTYA98PjfYQyu9DBwxdb2b2

- Instructions

| Instructions                          |  |
|---------------------------------------|--|
| #1 Token-2022 Program: Freeze Account |  |
| Account                               | diZa8NKEj7wTf31meXA9X2pyQnBsTnV4jsYUmwfFAK1  |
| FreezeAuthority                       | Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj |
| Mint                                  | JA6iL96LYav6GaMS1NnrZRTYA98PjfYQyu9DBwxdb2b2 |

- Account status

We can see that the status is passed to `Frozen`.

| Token-2022 Account |  |
|--------------------|--|
| Address            | diZa8NKEj7wTf31meXA9X2pyQnBsTnV4jsYUmwfFAK1              |
| Mint               | JA6iL96LYav6GaMS1NnrZRTYA98PjfYQyu9DBwxdb2b2             |
| Owner              | 9Grr8jKaZUji1VGj3uBBE3vwBmRbf48CGUchAUfxrqrk             |
| Token balance      | 300  |
| Status             | Frozen   |
| Extensions         | <code>immutableOwner</code> <code>pausableAccount</code> |

## Transfer

```
spl-token transfer [OPTIONS] \ <TOKEN_MINT_ADDRESS> <TOKEN_AMOUNT> \
<RECIPIENT_WALLET_ADDRESS or RECIPIENT_TOKEN_ACCOUNT_ADDRESS>
```

### Example

We try to transfer tokens from our user account to the admin account

```
solana config set -k $USER_SOLANA_KEYPAIR
spl-token transfer --program-2022 --from $USER_TOKEN_ACCOUNT $TOKEN_MINT 100
$ADMIN_TOKEN_ACCOUNT
```

## Result

Transfer 100 tokens

Sender: diZa8NKEj7wTf31meXA9X2pyQnBsTnV4jsYUmwfFAK1

Recipient: ApXcDVWCXhPgfAmUYqQ85JcYmyxqecqDPKXPz8B2fjqK

Error: Client(Error { request: Some(SendTransaction), kind: RpcError(RpcResponseError { code: -32002, message: "Transaction simulation failed: Error processing Instruction 0: custom program error: 0x11", data: }

SendTransactionPreflightFailure(RpcSimulateTransactionResult { err: }

Some(InstructionError(0, Custom(17))), logs: Some(["Program

TokenzQdBNbLqP5VEhdksAS6EPFLC1PHnBqCXEpPxuEb invoke [1]", "Program log:

Instruction: TransferChecked", "Program log: Error: **Account is frozen**", "Program

TokenzQdBNbLqP5VEhdksAS6EPFLC1PHnBqCXEpPxuEb consumed 1111 of 1111 compute

```
units", "Program TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb failed: custom  
program error: 0x11"], accounts: None, units_consumed: Some(1111),  
loaded_accounts_data_size: Some(684911), return_data: None, inner_instructions: None,  
replacement_blockhash: None } } } )
```

## Unfreeze (thaw)

```
spl-token thaw [OPTIONS] <TOKEN_ACCOUNT_ADDRESS>
```

```
solana config set -k $ADMIN_SOLANA_KEYPAIR  
spl-token thaw --freeze-authority $ADMIN_SOLANA_KEYPAIR $USER_TOKEN_ACCOUNT
```

## Result

Thawing account: diZa8NKEJ7wTf31meXA9X2pyQnBsTnV4jsYUmwfFAK1

Token: JA6iL96LYav6GaMS1NnrZRTYA98PjfYQyu9DBwxdb2b2

| Token-2022 Account |  |
|--------------------|--|
| Address            | diZa8NKEJ7wTf31meXA9X2pyQnBsTnV4jsYUmwfFAK1  |
| Mint               | JA6iL96LYav6GaMS1NnrZRTYA98PjfYQyu9DBwxdb2b2 |
| Owner              | 9Grr8jKaZUji1VGj3uBBE3vWBmRbf48CGUchAUfxrqk  |
| Token balance      | 300  |
| Status             | Initialized                                  |
| Extensions         | immutableOwner pausableAccount               |

Now we can perform our transfer:

```
solana config set -k $USER_SOLANA_KEYPAIR  
spl-token transfer --from $USER_TOKEN_ACCOUNT --owner $USER_SOLANA_KEYPAIR  
$TOKEN_MINT 100 $ADMIN_TOKEN_ACCOUNT
```

## Result

Transfer 100 tokens

Sender: diZa8NKEJ7wTf31meXA9X2pyQnBsTnV4jsYUmwfFAK1

Recipient: ApXcDVWCXhPgAmUYqQ85JcYmyxqecqDPKXPz8B2fjqK

| Token Balances                              |  |        |              |  |
|---|--|--------|--------------|--|
| ADDRESS                                     | TOKEN  | CHANGE | POST BALANCE |  |
| diZa8NKEJ7wTf31meXA9X2pyQnBsTnV4jsYUmwfFAK1 | JA6iL96LYav6GaMS1NnrZRTYA98PjfYQyu9DBwxdb2b2 | -100   | 200 tokens   |  |
| ApXcDVWCXhPgAmUYqQ85JcYmyxqecqDPKXPz8B2fjqK | JA6iL96LYav6GaMS1NnrZRTYA98PjfYQyu9DBwxdb2b2 | +100   | 1100 tokens  |  |

# Burn Tokens / Forced Transfer (Permanent Delegate)

The standard token version already allows a token holder to burn its own token

With Token-2022, it's possible to specify a permanent account delegate for a mint. This authority has unlimited delegate privileges over any account for that mint, meaning that it can burn or transfer any amount of tokens.

```
spl-token burn [OPTIONS] <TOKEN_ACCOUNT_ADDRESS> <TOKEN_AMOUNT>
```

```
spl-token burn $USER_TOKEN_ACCOUNT 5
```

Burn 5 tokens

Source: diZa8NKEJ7wTf31meXA9X2pyQnBsTnV4jsYUmwffFAK1

| Account Input(s) |  |              |                    |           |                 |
|------------------|--|--------------|--------------------|-----------|-----------------|
| #                | ADDRESS  | CHANGE (SOL) | POST BALANCE (SOL) | DETAILS   |                 |
| 1                | 9Grr8jKaZUji1VGj3uBBE3vWBmRbf48CGUchAUfxrqk<br>User                | ~@0.00005    | @999.99787308      | Fee Payer | Signer Writable |
| 2                | diZa8NKEJ7wTf31meXA9X2pyQnBsTnV4jsYUmwffFAK1<br>User Token Account | 0            | @0.00210192        | Writable  |                 |
| 3                | JA6iL96LYav6GaMS1NnrZRTYA98Pjf fyQyu9DBwxdb2b2<br>Mint Account     | 0            | @0.00414816        | Writable  |                 |
| 4                | Compute Budget Program   | 0            | @0.00000001        | Program   |                 |
| 5                | Token-2022 Program   | 0            | @0.00114144        | Program   |                 |

## Verification

```
spl-token balance --program-2022 --address $USER_TOKEN_ACCOUNT
```

## Burn as admin

```
solana config set -k $ADMIN_SOLANA_KEYPAIR
spl-token burn $USER_TOKEN_ACCOUNT 5
```

## Result

Burn 5 tokens

Source: diZa8NKEJ7wTf31meXA9X2pyQnBsTnV4jsYUmwffFAK1

The signer of the transaction will be the admin

| Account Input(s) |  |              |                    |           |                 |
|------------------|--|--------------|--------------------|-----------|-----------------|
| #                | ADDRESS  | CHANGE (SOL) | POST BALANCE (SOL) | DETAILS   |                 |
| 1                | Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj   | ~@0.00005    | @999.98953676      | Fee Payer | Signer Writable |
| 2                | diZa8NKEJ7wTf31meXA9X2pyQnBsTnV4jsYUmwffFAK1   | 0            | @0.00210192        | Writable  |                 |
| 3                | JA6iL96LYav6GaMS1NnrZRTYA98Pjf fyQyu9DBwxdb2b2 | 0            | @0.00414816        | Writable  |                 |
| 4                | Compute Budget Program                         | 0            | @0.00000001        | Program   |                 |
| 5                | Token-2022 Program                             | 0            | @0.00114144        | Program   |                 |

## User as signer

If I try to burn admin tokens with my user as the signer, I will have the error `InvalidAccountForFee` because my user does not have the delegate authority.

```
# Change the signer in the config for the user solana config
solana config set -k $USER_SOLANA_KEYPAIR
# try to burn admin token with the user as signer
spl-token burn $ADMIN_TOKEN_ACCOUNT 5
```

## Result

Burn 5 tokens

```
Source: ApXcDVWCXhPgfAmUYqQ85JcYmyxqecqDPKXPz8B2fjqK
Error: Client(Error { request: Some(SendTransaction), kind: RpcError(RpcResponseError {
code: -32002, message: "Transaction simulation failed: Error processing Instruction 0:
custom program error: 0x4", data:
SendTransactionPreflightFailure(RpcSimulateTransactionResult { err:
Some(InstructionError(0, Custom(4))), logs: Some(["Program
TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb invoke [1]", "Program log:
Instruction: BurnChecked", "Program log: Error: owner does not match", "Program
TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb consumed 2013 of 2013 compute
units", "Program TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb failed: custom
program error: 0x4"]), accounts: None, units_consumed: Some(2013),
loaded_accounts_data_size: Some(684673), return_data: None, inner_instructions: None,
replacement_blockhash: None }) } })})
```

## Forced transfer

A forced transfer is performed in the same way as a standard transfer

```
spl-token transfer [OPTIONS] <TOKEN_MINT_ADDRESS> \ <TOKEN_AMOUNT> \
<RECIPIENT_WALLET_ADDRESS or RECIPIENT_TOKEN_ACCOUNT_ADDRESS>
```

## Example

```
# Change the signer in the config for the admin solana config
solana config set -k $ADMIN_SOLANA_KEYPAIR
# forced transfer tokens from user -> admin with the admin signer key
spl-token transfer --program-2022 --from $USER_TOKEN_ACCOUNT $TOKEN_MINT 10
$ADMIN_TOKEN_ACCOUNT
```

## Result

Transfer 10 tokens

Sender: diZa8NKEJ7wTf31meXA9X2pyQnBsTnV4jsYUmwfFAK1

Recipient: ApXcDVWCXhPgfAmUYqQ85JcYmyxqecqDPKXPz8B2fjqK

| #1 Token-2022 Program: Transfer (Checked) |  | <a href="#">Raw</a> |
|---|--|---------------------|
| Authority                                 | <a href="#">Annh1c8T1nLkqU1dNNm7xUho7H6hU9KV6JE9THndHEfj</a> | Admin               |
| Destination                               | <a href="#">ApXcDVwCXhPgFAmUYqQ85JcYmyxqecqDPKXPz8B2fjqK</a> | Admin Token Account |
| Mint                                      | <a href="#">JA6iL96LYav6GaMS1NnrZRTYA98PjfYQyu9DBwdx2b2</a>  | Token Mint          |
| Source                                    | <a href="#">diZa8NKEJ7wTf31meXA9X2pyQnBsTnV4jsYUmwfFAK1</a>  | User Token Account  |
| Amount                                    | 10   |                     |

| Account Input(s) |  |               |                    |   |
|------------------|--|---------------|--------------------|---|
| #                | ADDRESS  | CHANGE (SOL)  | POST BALANCE (SOL) | DETAILS   |
| 1                | <a href="#">Annh1c8T1nLkqU1dNNm7xUho7H6hU9KV6JE9THndHEfj</a> | -@0.00013028  | @999.98825376      | <a href="#">Fee Payer</a> <a href="#">Signer</a> <a href="#">Writable</a> |
| 2                | <a href="#">JA6iL96LYav6GaMS1NnrZRTYA98PjfYQyu9DBwdx2b2</a>  | +\$0.00012528 | @0.00536616        | <a href="#">Writable</a>  |
| 3                | <a href="#">System Program</a>                               | 0             | @0.000000001       | <a href="#">Program</a>   |
| 4                | <a href="#">Compute Budget Program</a>                       | 0             | @0.000000001       | <a href="#">Program</a>   |
| 5                | <a href="#">Token-2022 Program</a>                           | 0             | @0.00114144        | <a href="#">Program</a>   |

## Pause / Resume All Activity

By enabling the pausable extension on your mint, the program aborts all transfers, mints, and burns when the `paused` flag is flipped.

This also includes force operation by the admin such as `burn` and `transfer`

### Pause

Pause

```
spl-token pause [OPTIONS] <TOKEN_MINT_ADDRESS>
spl-token pause $TOKEN_MINT
```

### Result

Pausing mint, burn, and transfer for JA6iL96LYav6GaMS1NnrZRTYA98PjfYQyu9DBwdx2b2

| #1 Token-2022 Program: Unknown Instruction |  | <a href="#">Raw</a>   |
|--|--|---|
| Program                                    |  | <a href="#">Token-2022 Program</a>  |
| Instruction Data (JSON)                    |  | {     "info": {         "authority": "Annh1c8T1nLkqU1dNNm7xUho7H6hU9KV6JE9THndHEfj",         "mint": "JA6iL96LYav6GaMS1NnrZRTYA98PjfYQyu9DBwdx2b2"     },     "type": "pause" } |

Trying a transfer will generate *Program log: Transferring, minting, and burning is paused on this mint*

```
solana config set -k $ADMIN_SOLANA_KEYPAIR
spl-token transfer $TOKEN_MINT 10 $USER_TOKEN_ACCOUNT
```

```
Transfer 10 tokens
Sender: ApXcDVWCXhPgfAmUYqQ85JcYmyxqecqDPKXPz8B2fjqK
Recipient: diZa8NKEj7wTf31meXA9X2pyQnBsTnV4jsYUmwfFAK1
Error: Client(Error { request: Some(SendTransaction), kind: RpcError(RpcResponseError {
code: -32002, message: "Transaction simulation failed: Error processing Instruction 0:
custom program error: 0x43", data:
SendTransactionPreflightFailure(RpcSimulateTransactionResult { err:
Some(InstructionError(0, Custom(67))), logs: Some(["Program
TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb invoke [1]", "Program log:
Instruction: TransferChecked", "Program log: Transferring, minting, and burning is
paused on this mint", "Program TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb
consumed 2192 of 2192 compute units", "Program
TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb failed: custom program error:
0x43"]), accounts: None, units_consumed: Some(2192), loaded_accounts_data_size:
Some(684911), return_data: None, inner_instructions: None, replacement_blockhash: None
}) } }) })
```

## Resume

Resume mint, burn, and transfer

```
spl-token resume [OPTIONS] <TOKEN_MINT_ADDRESS>
spl-token resume $TOKEN_MINT
```

Resuming mint, burn, and transfer for JA6iL96LYav6GaMS1NnrZRTYA98PjfYQyu9DBwxdb2b2



The screenshot shows the Token-2022 Program interface with the following details:

- #1 Token-2022 Program: Unknown Instruction**
- Program**: spl-token resume \$TOKEN\_MINT
- Instruction Data (JSON)**:

```
{
  "info": {
    "authority": "Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj",
    "mint": "JA6iL96LYav6GaMS1NnrZRTYA98PjfYQyu9DBwxdb2b2"
  },
  "type": "resume"
}
```
- Raw** button

## Update On-Chain Metadata

To facilitate token-metadata usage, Token-2022 allows a mint creator to include their token's metadata directly in the mint account.

```
spl-token update-metadata <TOKEN_MINT_ADDRESS> <FIELD_NAME>
```

## Example

```

# Update name, symbol, or URI
spl-token update-metadata $TOKEN_MINT name "New
CMTAT Token"
spl-token update-metadata $TOKEN_MINT symbol "NEW"
spl-token update-metadata $TOKEN_MINT uri "https://cmta.ch/token_new.json"
# Add custom compliance fields
spl-token update-metadata $TOKEN_MINT TermsUri "https://cmta.ch/token_new.pdf"
spl-token update-metadata $TOKEN_MINT termsHash
"0x9c22ff5f21f0b81b113e63f7db6da94fefef11b2119b4088b89664fb9a3cb658"
spl-token update-metadata $TOKEN_MINT jurisdiction "EU-MiCA"
spl-token update-metadata $TOKEN_MINT issuer "CMTA"

```

## Result

```
spl-token display $TOKEN_MINT
```

spl-token display \$TOKEN\_MINT

SPL Token Mint

Address: JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2

Program: TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb

Supply: 1270

Decimals: 0

Mint authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Freeze authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Extensions

Close authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Permanent delegate: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Metadata Pointer:

Authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Metadata address: JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2

Metadata:

Update Authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Mint: JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2

Name: CMTATTOKEN

Symbol: NEW

URI: [https://cmta.ch/token\\_new.json](https://cmta.ch/token_new.json)

TermsUri: [https://cmta.ch/token\\_new.pdf](https://cmta.ch/token_new.pdf)

termsHash: 0x9c22ff5f21f0b81b113e63f7db6da94fefef11b2119b4088b89664fb9a3cb658

jurisdiction: EU-MiCA

issuer: CMTA

#2 Token-2022 Program: Unknown Instruction

[Raw](#)

|         |                    |
|---------|--------------------|
| Program | Token-2022 Program |
|---------|--------------------|

Instruction Data (JSON)

```
{
  "info": {
    "field": "issuer",
    "metadata": "JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2",
    "updateAuthority": "Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj",
    "value": "CMTA"
  },
  "type": "updateTokenMetadataField"
}
```

## Remove a custom field

```
spl-token update-metadata $TOKEN_MINT termsHash --remove
```

### Result

```
spl-token display $TOKEN_MINT
```

SPL Token Mint

Address: JA6iL96LYav6GaMS1NnrZRTYA98PjfYQyu9DBwxdb2b2

Program: TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb

Supply: 1270

Decimals: 0

Mint authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Freeze authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Extensions

Close authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Permanent delegate: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Metadata Pointer:

Authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Metadata address: JA6iL96LYav6GaMS1NnrZRTYA98PjfYQyu9DBwxdb2b2

Metadata:

Update Authority: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

Mint: JA6iL96LYav6GaMS1NnrZRTYA98PjfYQyu9DBwxdb2b2

Name: CMTATTOKEN

Symbol: NEW

URI: [https://cmta.ch/token\\_new.json](https://cmta.ch/token_new.json)

TermsUri: [https://cmta.ch/token\\_new.pdf](https://cmta.ch/token_new.pdf)

Jurisdiction: EU-MiCA

Issuer: CMTA

#1 Token-2022 Program: Unknown Instruction

Program

Instruction Data (JSON)

```
{ "info": { "idempotent": true, "key": "termsHash", "metadata": "JA6iL96LYav6GaMS1NnrZRTYA98PjfYQyu9DBwxdb2b2", "updateAuthority": "Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj" }, "type": "removeTokenMetadataKey" }
```

Raw

Token-2022 Program

## Deactivate token

### Burn all tokens

Using the delegate authorities, burn all remaining tokens as seen above.

```
spl-token balance --address $USER_TOKEN_ACCOUNT
```

Result: 150

```
spl-token burn $USER_TOKEN_ACCOUNT 150
```

Burn 150 tokens

```
spl-token balance --address $ADMIN_TOKEN_ACCOUNT
```

Result: 1120

```
spl-token burn $ADMIN_TOKEN_ACCOUNT 1120
```

```
spl-token supply $TOKEN_MINT
```

Result: 0

## Deactivate Authorities

Then you can optionally deactivate all authorities, except the `MintCloseAuthority`

```
spl-token authorize [OPTIONS] <TOKEN_ADDRESS> <AUTHORITY_TYPE> [--]
[AUTHORITY_ADDRESS]
```

### Example

```
spl-token authorize $TOKEN_MINT mint --disable
spl-token authorize $TOKEN_MINT freeze --disable
spl-token authorize $TOKEN_MINT pause --disable
```

### Result

Updating JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2

Current mint: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

New mint: disabled

Updating JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2

Current freeze: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

New freeze: disabled

Updating JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2

Current pause: Annh1c8T1nLKqU1dNNm7xUho7H6hU9KV6JE9THndHEfj

New pause: disabled

## Close the Mint

The Token program allows owners to close token accounts, but it is impossible to close mint accounts. In Token-2022, it is possible to close mints by initializing the `MintCloseAuthority` extension before initializing the mint.

The Admin must burn all tokens before closing the token mint

```
spl-token close-mint $TOKEN_MINT
```

If you try to mint new tokens:

```
spl-token mint --program-2022 --mint-authority $ADMIN_SOLANA_KEYPAIR  
$TOKEN_MINT 300 $USER_TOKEN_ACCOUNT
```

This will generate the following error:

```
Error: "Account JA6iL96LYav6GaMS1NnrZRTYA98PjfyQyu9DBwxdb2b2 not found"
```

## CMAT with whitelist

CMTAT Solidity version allows to restrict transactions to a predefined list of approved wallet addresses, known as a **whitelist/allowlist**. This is done through a specific deployment version (`CMTAT Allowlist`) or through the RuleEngine (e.g. `RuleWhitelist`).

On Solana, this is done by enabling the extension `Default Account State` at the creation of the token.

## Command line

With the command-line, this is done by using the following option `--default-account-state` with the argument `initialized` or `frozen`.

```
spl-token create-token \  --program-id  
TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb \  --decimals <decimals>\  --mint-  
authority < Solana Keypair> \  --enable-permanent-delegate \  --enable-pause \  
--enable-close \  --enable-metadata  --enable-freeze --default-account-state  
frozen
```

- With `initialized`, the frozen status by default is not enabled for the moment, but can be enabled later by the authorized authority.
- With `frozen`, all accounts are frozen by default and required to be unfrozen to allow to be a token holder.

Over time, if the mint creator decides to relax this restriction, the freeze authority may sign an `update_default_account_state` instruction to make all accounts unfrozen by default.

```
spl-token update-default-account-state [OPTIONS] <TOKEN_MINT_ADDRESS> <STATE>
```

## Example

| Label         | Value  |
|---------------|--|
| Admin address | 9G3BN5Jeemo5nQbtPNhcKyeyithQZ3kti8ARYYD7yEQp |
| User address  | DcDkMbpHDSUGXwFroZgE6chGPXFjcL4qr1YReViaMmrL |

## Create token mint

### Command

```
spl-token create-token --program-id TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb  
--decimals 0 --mint-authority $ADMIN_SOLANA_KEYPAIR --enable-permanent-  
delegate --enable-pause --enable-close --enable-metadata --enable-freeze --  
default-account-state frozen
```

### Result

Creating token AJCDqXP16eJByoZ8gNono6Lm8VzaPHsBganJci1CF5tQ under program TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb

To initialize metadata inside the mint, please run `spl-token initialize-metadata`

`AJCDqXP16eJByoZ8gNono6Lm8VzaPHsBganJci1CF5tQ <YOUR_TOKEN_NAME>`

`<YOUR_TOKEN_SYMBOL> <YOUR_TOKEN_URI>`, and sign with the mint authority.

Address: AJCDqXP16eJByoZ8gNono6Lm8VzaPHsBganJci1CF5tQ

Decimals: 0

| Token-2022 Mint  |   |
|------------------|---|
| Address          | <a href="#">AJCDqXP16eJByoZ8gNono6Lm8VzaPHsBganJci1CF5tQ</a>  |
| Current Supply   | 0   |
| Mint Authority   | <a href="#">9G3BN5Jeemo5nQbtPNhcKyeyitHQZ3kti8ARYYD7yEqp</a>  |
| Freeze Authority | <a href="#">9G3BN5Jeemo5nQbtPNhcKyeyitHQZ3kti8ARYYD7yEqp</a>  |
| Decimals         | 0   |
| Extensions       | <a href="#">mintCloseAuthority</a> <a href="#">defaultAccountState</a> <a href="#">permanentDelegate</a> <a href="#">metadataPointer</a> <a href="#">pausableConfig</a> |

In the explorer, we can see that our token mint has the extension `defaultAccountState`

We export our token address in our bash environment

```
export TOKEN_MINT=AJCDqXP16eJByoZ8gNono6Lm8VzaPHsBganJci1CF5tQ
```

## Create token account

We create then the token account for the admin

```
spl-token create-account --program-2022 $TOKEN_MINT --owner  
$ADMIN_SOLANA_KEYPAIR
```

Creating account DRFWi78CJhhcfca9yZCTZbaBjnsTWhiP2yLiu6oVmrxG

### Instruction details

|                         |   |                     |
|-------------------------|---|---------------------|
| #2                      | Token-2022 Program: Initialize Mint Close Authority   | <a href="#">Raw</a> |
| Mint                    | <a href="#">AJCDqXP16eJByoZ8gNono6Lm8VzaPHsBganJci1CF5tQ</a>  |                     |
| NewAuthority            | <a href="#">9G3BN5Jeemo5nQbtPNhcKyeyitHQZ3kti8ARYYD7yEqP</a>  |                     |
| #3                      | Token-2022 Program: Initialize Permanent Delegate   | <a href="#">Raw</a> |
| Delegate                | <a href="#">9G3BN5Jeemo5nQbtPNhcKyeyitHQZ3kti8ARYYD7yEqP</a>  |                     |
| Mint                    | <a href="#">AJCDqXP16eJByoZ8gNono6Lm8VzaPHsBganJci1CF5tQ</a>  |                     |
| #4                      | Token-2022 Program: Unknown Instruction   | <a href="#">Raw</a> |
| Program                 | <a href="#">Token-2022_Program</a>  |                     |
| Instruction Data (JSON) | {<br>"info": {<br>"accountState": "frozen",<br>"mint": "AJCDqXP16eJByoZ8gNono6Lm8VzaPHsBganJci1CF5tQ"<br>},<br>"type": "initializeDefaultAccountState"<br>} |                     |
| #5                      | Token-2022 Program: Initialize Metadata Pointer   | <a href="#">Raw</a> |
| Authority               | <a href="#">9G3BN5Jeemo5nQbtPNhcKyeyitHQZ3kti8ARYYD7yEqP</a>  |                     |
| MetadataAddress         | <a href="#">AJCDqXP16eJByoZ8gNono6Lm8VzaPHsBganJci1CF5tQ</a>  |                     |
| Mint                    | <a href="#">AJCDqXP16eJByoZ8gNono6Lm8VzaPHsBganJci1CF5tQ</a>  |                     |

## Account status

We can see that the status is `frozen`

|                    |  |  |
|--------------------|--|--|
| Token-2022 Account |  | <a href="#">Refresh</a>                                  |
| Address            | <a href="#">DRFWi78CJhhcfca9yZCTZbaBjnsTWhiP2yLiu6oVmrxG</a> |  |
| Mint               | <a href="#">AJCDqXP16eJByoZ8gNono6Lm8VzaPHsBganJci1CF5tQ</a> |  |
| Owner              | <a href="#">9G3BN5Jeemo5nQbtPNhcKyeyitHQZ3kti8ARYYD7yEqP</a> |  |
| Token balance      | 0  |  |
| Status             |  | <code>Frozen</code>                                      |
| Extensions         |  | <code>immutableOwner</code> <code>pausableAccount</code> |

We export the token account address

```
export ADMIN_TOKEN_ACCOUNT=DRFWi78CJhhcfca9yZCTZbaBjnsTWhiP2yLiu6oVmrxG
```

## Thaw / unfreeze token account

```
solana config set -k $ADMIN_SOLANA_KEYPAIR
spl-token thaw --freeze-authority $ADMIN_SOLANA_KEYPAIR $ADMIN_TOKEN_ACCOUNT
```

Thawing account: DRFWi78CJhhcfca9yZCTZbaBjnsTWhiP2yLiu6oVmrxG  
 Token: AJCDqXP16eJByoZ8gNono6Lm8VzaPHsBganJci1CF5tQ

## New status

| Token-2022 Account |  |
|--------------------|--|
| Address            | DRFWi78CJhhcfca9yZCTZbaBjnsTWh1P2yLiu6oVmrxG |
| Mint               | AJCDqXP16eJByoZ8gNomo6Lm8VzaPHsBganJci1CF5tQ |
| Owner              | 9G3BN5Jeemo5nQbtPNhcKyeyitHQZ3kti8ARYYD7yEQp |
| Token balance      | 0  |
| Status             | Initialized                                  |
| Extensions         | immutableOwner pausableAccount               |

## Update token mint

```
spl-token update-default-account-state $TOKEN_MINT initialized
```

Account Input(s)

| # | ADDRESS                                      | CHANGE (SOL) | POST BALANCE (SOL)   | DETAILS                   |
|---|--|--------------|----------------------|---------------------------|
| 1 | 9G3BN5Jeemo5nQbtPNhcKyeyitHQZ3kti8ARYYD7yEQp | -@0.000065   | @499,999,999.9945651 | Fee Payer Signer Writable |
| 2 | AJCDqXP16eJByoZ8gNomo6Lm8VzaPHsBganJci1CF5tQ | 0            | @0.00331296          | Writable                  |
| 3 | Compute Budget Program                       | 0            | @0.000000001         | Program                   |
| 4 | Token-2022 Program                           | 0            | @0.00114144          | Program                   |

Instructions

#1 Token-2022 Program: Unknown Instruction

Program

Instruction Data (JSON)

```
{  
  "info": {  
    "accountState": "initialized",  
    "freezeAuthority": "9G3BN5Jeemo5nQbtPNhcKyeyitHQZ3kti8ARYYD7yEQp",  
    "mint": "AJCDqXP16eJByoZ8gNomo6Lm8VzaPHsBganJci1CF5tQ"  
  },  
  "type": "updateDefaultAccountState"  
}
```

## Create token account

```
solana config set -k $ADMIN_SOLANA_KEYPAIR  
spl-token create-account --program-2022 $TOKEN_MINT --owner  
$USER_SOLANA_KEYPAIR
```

Creating account 4SKEr1QXpy9H5KV8voAM1FjzPoxAjs26WfenkgPXdq7E

We can see in the explorer that the status of the new created account is `initialized` instead of `frozen`

| Token-2022 Account |  |
|--------------------|--|
| Address            | 4SKEr1QXpy9H5KV8voAM1FjzPoxAjs26WfenkgPXdq7E             |
| Mint               | AJCDqXP16eJByoZ8gNono6Lm8VZaPHsBganJci1CF5tQ             |
| Owner              | DcDkMbpHDSUGXwFroZgE6chGPXFjcL4qr1YReViaMmrL             |
| Token balance      | 0  |
| Status             | Initialized  |
| Extensions         | <span>immutableOwner</span> <span>pausableAccount</span> |

## Reference

---

- [Token](#)
- [Solana Program - Token-2022](#)
- [Solana - Getting Started with Token Extensions](#)
- [Solana Program - Extension Guide](#)
- [Solana - Issuing Tokenized Equities on Solana Report](#)