Create Coins and Tokens

Coins and tokens on Sui are similar. In practice, the terms are used interchangeably, but there are some differences in their implementation. You can learn about these differences in the respective standard documentation, Closed-Loop Token and Coin .

Publishing a coin on Sui is similar to publishing a new type. The main difference is the requirement of a one time witness when creating a coin.

The Coin is a generic implementation of a coin on Sui. Access to the TreasuryCap provides control over the minting and burning of coins. Further transactions can be sent directly to the sui::coin::Coin with TreasuryCap object as authorization.

The example module includes a mint function. You pass the TreasuryCap created from the init function to the module's mint function. The function then uses the mint function from the Coin module to create (mint) a coin and then transfer it to an address.

If you published the previous example to a Sui network, you can use the sui client call command to mint coins and deliver them to the address you provide. See Sui CLI for more information on the command line interface.

Beginning with the Sui v1.24.1 release , the --gas-budget option is no longer required for CLI commands.

If the call is successful your console displays the result, which includes a Balance Changes section with the following information included:

The Sui framework provides a DenyList singleton, shared object that the bearer of a DenyCapV2 can access to specify a list of addresses that are unable to use a Sui core type. The initial use case for DenyList , however, focuses on limiting access to coins of a specified type. This is useful, for example, when creating a regulated coin on Sui that requires the ability to block certain addresses from using it as inputs to transactions. Regulated coins on Sui satisfy any regulations that require the ability to prevent known bad actors from having access to those coins.

The DenyList object is a system object that has the address 0x403 . You cannot create it yourself.

If you need the ability to deny specific addresses from having access to your coin, you can use the create_regulated_currency_v2 function (instead of create_currency) to create it.

Behind the scenes, create_regulated_currency_v2 uses the create_currency function to create the coin, but also produces a DenyCapV2 object that allows its bearer to control access to the coin's deny list in a DenyList object. Consequently, the way to create a coin using create_regulated_currency_v2 is similar to the previous example, with the addition of a transfer of the DenyCap object to the module publisher.

Tokens reuse the TreasuryCap defined in the sui::coin module and therefore have the same initialization process. The coin::create_currency function guarantees the uniqueness of the TreasuryCap and forces the creation of a CoinMetadata object.

Coin-like functions perform the minting and burning of tokens. Both require the TreasuryCap :

See Closed-Loop Token standard for complete details of working with tokens.

DenyList

The Sui framework provides a DenyList singleton, shared object that the bearer of a DenyCapV2 can access to specify a list of addresses that are unable to use a Sui core type. The initial use case for DenyList , however, focuses on limiting access to coins of a specified type. This is useful, for example, when creating a regulated coin on Sui that requires the ability to block certain addresses from using it as inputs to transactions. Regulated coins on Sui satisfy any regulations that require the ability to prevent known bad actors from having access to those coins.

The DenyList object is a system object that has the address 0x403 . You cannot create it yourself.

If you need the ability to deny specific addresses from having access to your coin, you can use the create_regulated_currency_v2 function (instead of create_currency) to create it.

Behind the scenes, create_regulated_currency_v2 uses the create_currency function to create the coin, but also produces a DenyCapV2 object that allows its bearer to control access to the coin's deny list in a DenyList object. Consequently, the way to create a coin using create_regulated_currency_v2 is similar to the previous example, with the addition of a transfer of the DenyCap object to the module publisher.

Tokens reuse the TreasuryCap defined in the sui::coin module and therefore have the same initialization process. The coin::create_currency function guarantees the uniqueness of the TreasuryCap and forces the creation of a CoinMetadata object.

Coin-like functions perform the minting and burning of tokens. Both require the TreasuryCap :

See Closed-Loop Token standard for complete details of working with tokens.

Create regulated coin

If you need the ability to deny specific addresses from having access to your coin, you can use the create_regulated_currency_v2 function (instead of create_currency) to create it.

Behind the scenes, create_regulated_currency_v2 uses the create_currency function to create the coin, but also produces a DenyCapV2 object that allows its bearer to control access to the coin's deny list in a DenyList object. Consequently, the way to create a coin using create_regulated_currency_v2 is similar to the previous example, with the addition of a transfer of the DenyCap object to the module publisher.

Tokens reuse the TreasuryCap defined in the sui::coin module and therefore have the same initialization process. The coin::create_currency function guarantees the uniqueness of the TreasuryCap and forces the creation of a CoinMetadata object.

Coin-like functions perform the minting and burning of tokens. Both require the TreasuryCap :

See Closed-Loop Token standard for complete details of working with tokens.

Create tokens

Tokens reuse the TreasuryCap defined in the sui::coin module and therefore have the same initialization process. The coin::create_currency function guarantees the uniqueness of the TreasuryCap and forces the creation of a CoinMetadata object.

Coin-like functions perform the minting and burning of tokens. Both require the TreasuryCap :

See Closed-Loop Token standard for complete details of working with tokens.

Related links