Coin Standard

The Coin standard is the technical standard used for smart contracts on Sui for creating coins on the Sui blockchain. The standardization of coin creation on Sui means that wallets, exchanges, and other smart contracts can manage coins created on Sui the same as they manage SUI, without any additional processing logic.

See Sui Tokenomics to learn more about the SUI native coin and its use on the Sui network.

Although coins on Sui follow the Coin standard, they can offer specialized abilities. For example, you can create a regulated token that allows its creator to add specific addresses to a deny list, so that the identified addresses cannot use the token as inputs to transactions.

See the coin module documentation for all available options when creating a coin-type token on Sui.

In the Sui blockchain ecosystem, the Coin type represents open-loop fungible tokens (see Token for closed-loop tokens). Coins are denominated by their type parameter, T , which is also associated with metadata (like name, symbol, decimal precision, and so on) that applies to all instances of Coin . The sui::coin module exposes an interface over Coin that treats it as fungible, meaning that a unit of T held in one instance of Coin is interchangeable with any other unit of T , much like how traditional fiat currencies operate.

The documentation refers to fungible tokens created on Sui using the Coin standard as "coins". For fungible tokens created on Sui using the Closed-Loop Token standard , the documentation uses the term "tokens". In practice, the terms for both these objects are often interchangeable.

When you create a coin using the coin::create_currency function, the publisher of the smart contract that creates the coin receives a TreasuryCap object. The TreasuryCap object is required to mint new coins or to burn current ones. Consequently, only addresses that have access to this object are able to maintain the coin supply on the Sui network.

The TreasuryCap object is transferable, so a third party can take over the management of a coin that you create if you transfer the TreasuryCap . After transferring the capability, however, you are no longer able to mint and burn tokens yourself.

The Coin standard includes the ability to create regulated coins. To create a regulated coin, you use the coin::create_regulated_currency_v2 function (which uses the coin::create_currency function itself), but which also returns a DenyCap capability. The DenyCap capability allows the bearer to maintain a list of addresses that aren't allowed to use the token.

The regulated-coin-sample repository provides an example of regulated coin creation.

The list of addresses that aren't able to use a particular regulated coin is held within a system-created DenyList shared object. If you have access to the DenyCap , then you can use the coin::deny_list_v2_add and coin::deny_list_v2_remove functions to add and remove addresses.

Regulated coin objects include an allow_global_pause Boolean field. When set to true , the bearer of the DenyCapV2 object for the coin type can use the coin::deny_list_v2_enable_global_pause function to pause coin activity indefinitely. Immediately upon the bearer initiating the pause, the network disallows the coin type as input for any transactions. At the start of the next epoch (epochs last ~24 hours), the network additionally disallows all addresses from receiving the coin type.

When the bearer of the DenyCapV2 object for the coin type removes the pause using coin::deny_list_v2_disable_global_pause , the coins are immediately available to use again as transaction inputs. Addresses cannot receive the coin type, however, until the following epoch.

The global pause functionality does not affect the deny list for the coin. After clearing the pause for the coin, any addresses included in the deny list are still unable to interact with the coin.

Each coin you create includes metadata that describes it. Typically, smart contracts freeze this object upon creation using the transfer::public_freeze_object function because the metadata for coins should almost never change. Regulated coins freeze the metadata they create automatically.

Regular coins using the Coin standard include a CoinMetadata object. As mentioned previously, regulated coins build on top of the same procedure that creates regular coins, so they receive the same metadata object in addition to a RegulatedCoinMetadata object that includes deny list information.

The fields of the metadata objects include the following:

The coin module provides the logic for creating and destroying coins on the Sui network (as long as you own the associated

TreasuryCap). These functions are the same for all coins and each requires the TreasuryCap as an input.

Use the coin::mint function to create new tokens.

The signature shows that a Coin results from calling the function with a TreasuryCap , value for the coin created, and the transaction context. The function updates the total supply in TreasuryCap automatically. Upon display, the coin value respects the decimals value in the metadata. So, if you supply 1000000 as the coin value that has a decimal value of 6 , the coin's value displays as 1.000000 .

Use the coin::burn function to destroy current tokens.

The signature shows that only the TreasuryCap and coin object you want to burn are necessary inputs, returning the amount by which the supply was decreased (value of the coin). The function does not allow you to burn more coins than are available in the supply.

The deny list is only applicable to regulated coins. As mentioned previously, when you create a regulated coin you receive a DenyCapV2 that authorizes the bearer to add and remove addresses from the system-created DenyList object. Any address on the list for your coin is unable to use the coin as an input to transactions starting immediately upon being added. At the epoch that follows address addition to the deny list, the addresses additionally cannot receive the coin type. In other words, an address that gets added to the deny list for a coin type is immediately unable to send the coin. At the start of the following epoch, the address is still unable to send the coin but is also unable to receive it. From that point, the address cannot interact with the coin until expressly removed from the deny list by the DenyCapV2 bearer.

Use the coin::deny_list_v2_add function to add the provided address to the deny list for your coin. The signature for the function is:

When using this function, you provide the DenyList object (0x403), the DenyCap you receive on coin creation, the address to add to the list, and the transaction context. After using this function, the address you provide is unable to use your coin by the next epoch.

Use the coin::deny_list_v2_remove function to remove addresses from the deny list for your coin.

When using this function, you provide the DenyList object (0x403), the DenyCapV2 you receive on coin creation, the address to remove from the list, and the transaction context. If you try to remove an address that isn't on the list, you receive an ENotFrozen error and the function aborts. After calling this function, the address you provide is able to use your coin by the next epoch.

You can use either the TypeScript or Rust SDK to manipulate the addresses held in the DenyList for your coin. The following examples are based on the regulated coin sample .

Globally pausing coin activity is only applicable to regulated coin types.

To pause activity across the network for a regulated coin type with the allow_global_pause field set to true , use coin::deny_list_v2_enable_global_pause . You must provide the DenyCapV2 object for the coin type to initiate the pause. Transaction activity is paused immediately, and no addresses can receive the coin in the epoch that follows the call to pause.

To restart network activity for a paused regulated coin, use the coin::deny_list_v2_disable_global_pause function. As with pausing, you must provide the DenyCapV2 object for the coin type. Transaction activity resumes immediately, and addresses can begin receiving the coin in the epoch that follows the call to remove the pause.

You can use the following functions to retrieve data from coins.

Use the following functions to get the values for the respective fields on the metadata object for coins.

Use the coin::supply function to get the current supply of a given coin.

If the CoinMetadata object was not frozen upon creation, you can use the following functions to update its values.

Each function signature is similar. Replace and with the values defined in the table to get the signature of each function:

RegulatedCoinMetadata is frozen upon creation, so there are no functions to update its data.

Check out the following content for more information about coins and tokens on Sui:

Fungible tokens

In the Sui blockchain ecosystem, the Coin type represents open-loop fungible tokens (see Token for closed-loop tokens). Coins are

denominated by their type parameter, T , which is also associated with metadata (like name, symbol, decimal precision, and so on) that applies to all instances of Coin . The sui::coin module exposes an interface over Coin that treats it as fungible, meaning that a unit of T held in one instance of Coin is interchangeable with any other unit of T , much like how traditional fiat currencies operate.

The documentation refers to fungible tokens created on Sui using the Coin standard as "coins". For fungible tokens created on Sui using the Closed-Loop Token standard , the documentation uses the term "tokens". In practice, the terms for both these objects are often interchangeable.

When you create a coin using the coin::create_currency function, the publisher of the smart contract that creates the coin receives a TreasuryCap object. The TreasuryCap object is required to mint new coins or to burn current ones. Consequently, only addresses that have access to this object are able to maintain the coin supply on the Sui network.

The TreasuryCap object is transferable, so a third party can take over the management of a coin that you create if you transfer the TreasuryCap . After transferring the capability, however, you are no longer able to mint and burn tokens yourself.

The Coin standard includes the ability to create regulated coins. To create a regulated coin, you use the coin::create_regulated_currency_v2 function (which uses the coin::create_currency function itself), but which also returns a DenyCap capability. The DenyCap capability allows the bearer to maintain a list of addresses that aren't allowed to use the token.

The regulated-coin-sample repository provides an example of regulated coin creation.

The list of addresses that aren't able to use a particular regulated coin is held within a system-created DenyList shared object. If you have access to the DenyCap , then you can use the coin::deny_list_v2_add and coin::deny_list_v2_remove functions to add and remove addresses.

Regulated coin objects include an allow_global_pause Boolean field. When set to true , the bearer of the DenyCapV2 object for the coin type can use the coin::deny_list_v2_enable_global_pause function to pause coin activity indefinitely. Immediately upon the bearer initiating the pause, the network disallows the coin type as input for any transactions. At the start of the next epoch (epochs last ~24 hours), the network additionally disallows all addresses from receiving the coin type.

When the bearer of the DenyCapV2 object for the coin type removes the pause using coin::deny_list_v2_disable_global_pause , the coins are immediately available to use again as transaction inputs. Addresses cannot receive the coin type, however, until the following epoch.

The global pause functionality does not affect the deny list for the coin. After clearing the pause for the coin, any addresses included in the deny list are still unable to interact with the coin.

Each coin you create includes metadata that describes it. Typically, smart contracts freeze this object upon creation using the transfer::public_freeze_object function because the metadata for coins should almost never change. Regulated coins freeze the metadata they create automatically.

Regular coins using the Coin standard include a CoinMetadata object. As mentioned previously, regulated coins build on top of the same procedure that creates regular coins, so they receive the same metadata object in addition to a RegulatedCoinMetadata object that includes deny list information.

The fields of the metadata objects include the following:

The coin module provides the logic for creating and destroying coins on the Sui network (as long as you own the associated TreasuryCap). These functions are the same for all coins and each requires the TreasuryCap as an input.

Use the coin::mint function to create new tokens.

The signature shows that a Coin results from calling the function with a TreasuryCap , value for the coin created, and the transaction context. The function updates the total supply in TreasuryCap automatically. Upon display, the coin value respects the decimals value in the metadata. So, if you supply 1000000 as the coin value that has a decimal value of 6 , the coin's value displays as 1.000000 .

Use the coin::burn function to destroy current tokens.

The signature shows that only the TreasuryCap and coin object you want to burn are necessary inputs, returning the amount by which the supply was decreased (value of the coin). The function does not allow you to burn more coins than are available in the supply.

The deny list is only applicable to regulated coins. As mentioned previously, when you create a regulated coin you receive a

DenyCapV2 that authorizes the bearer to add and remove addresses from the system-created DenyList object. Any address on the list for your coin is unable to use the coin as an input to transactions starting immediately upon being added. At the epoch that follows address addition to the deny list, the addresses additionally cannot receive the coin type. In other words, an address that gets added to the deny list for a coin type is immediately unable to send the coin. At the start of the following epoch, the address is still unable to send the coin but is also unable to receive it. From that point, the address cannot interact with the coin until expressly removed from the deny list by the DenyCapV2 bearer.

Use the coin::deny_list_v2_add function to add the provided address to the deny list for your coin. The signature for the function is:

When using this function, you provide the DenyList object (0x403), the DenyCap you receive on coin creation, the address to add to the list, and the transaction context. After using this function, the address you provide is unable to use your coin by the next epoch.

Use the coin::deny_list_v2_remove function to remove addresses from the deny list for your coin.

When using this function, you provide the DenyList object (0x403), the DenyCapV2 you receive on coin creation, the address to remove from the list, and the transaction context. If you try to remove an address that isn't on the list, you receive an ENotFrozen error and the function aborts. After calling this function, the address you provide is able to use your coin by the next epoch.

You can use either the TypeScript or Rust SDK to manipulate the addresses held in the DenyList for your coin. The following examples are based on the regulated coin sample .

Globally pausing coin activity is only applicable to regulated coin types.

To pause activity across the network for a regulated coin type with the allow_global_pause field set to true , use coin::deny_list_v2_enable_global_pause . You must provide the DenyCapV2 object for the coin type to initiate the pause. Transaction activity is paused immediately, and no addresses can receive the coin in the epoch that follows the call to pause.

To restart network activity for a paused regulated coin, use the coin::deny_list_v2_disable_global_pause function. As with pausing, you must provide the DenyCapV2 object for the coin type. Transaction activity resumes immediately, and addresses can begin receiving the coin in the epoch that follows the call to remove the pause.

You can use the following functions to retrieve data from coins.

Use the following functions to get the values for the respective fields on the metadata object for coins.

Use the coin::supply function to get the current supply of a given coin.

If the CoinMetadata object was not frozen upon creation, you can use the following functions to update its values.

Each function signature is similar. Replace and with the values defined in the table to get the signature of each function:

RegulatedCoinMetadata is frozen upon creation, so there are no functions to update its data.

Check out the following content for more information about coins and tokens on Sui:

Treasury capability

When you create a coin using the coin::create_currency function, the publisher of the smart contract that creates the coin receives a TreasuryCap object. The TreasuryCap object is required to mint new coins or to burn current ones. Consequently, only addresses that have access to this object are able to maintain the coin supply on the Sui network.

The TreasuryCap object is transferable, so a third party can take over the management of a coin that you create if you transfer the TreasuryCap . After transferring the capability, however, you are no longer able to mint and burn tokens yourself.

The Coin standard includes the ability to create regulated coins. To create a regulated coin, you use the coin::create_regulated_currency_v2 function (which uses the coin::create_currency function itself), but which also returns a DenyCap capability. The DenyCap capability allows the bearer to maintain a list of addresses that aren't allowed to use the token.

The regulated-coin-sample repository provides an example of regulated coin creation.

The list of addresses that aren't able to use a particular regulated coin is held within a system-created DenyList shared object. If you have access to the DenyCap , then you can use the coin::deny_list_v2_add and coin::deny_list_v2_remove functions to add and remove addresses.

Regulated coin objects include an allow_global_pause Boolean field. When set to true , the bearer of the DenyCapV2 object for the coin type can use the coin::deny_list_v2_enable_global_pause function to pause coin activity indefinitely. Immediately upon the bearer initiating the pause, the network disallows the coin type as input for any transactions. At the start of the next epoch (epochs last ~24 hours), the network additionally disallows all addresses from receiving the coin type.

When the bearer of the DenyCapV2 object for the coin type removes the pause using coin::deny_list_v2_disable_global_pause , the coins are immediately available to use again as transaction inputs. Addresses cannot receive the coin type, however, until the following epoch.

The global pause functionality does not affect the deny list for the coin. After clearing the pause for the coin, any addresses included in the deny list are still unable to interact with the coin.

Each coin you create includes metadata that describes it. Typically, smart contracts freeze this object upon creation using the transfer::public_freeze_object function because the metadata for coins should almost never change. Regulated coins freeze the metadata they create automatically.

Regular coins using the Coin standard include a CoinMetadata object. As mentioned previously, regulated coins build on top of the same procedure that creates regular coins, so they receive the same metadata object in addition to a RegulatedCoinMetadata object that includes deny list information.

The fields of the metadata objects include the following:

The coin module provides the logic for creating and destroying coins on the Sui network (as long as you own the associated TreasuryCap). These functions are the same for all coins and each requires the TreasuryCap as an input.

Use the coin::mint function to create new tokens.

The signature shows that a Coin results from calling the function with a TreasuryCap , value for the coin created, and the transaction context. The function updates the total supply in TreasuryCap automatically. Upon display, the coin value respects the decimals value in the metadata. So, if you supply 1000000 as the coin value that has a decimal value of 6 , the coin's value displays as 1.000000 .

Use the coin::burn function to destroy current tokens.

The signature shows that only the TreasuryCap and coin object you want to burn are necessary inputs, returning the amount by which the supply was decreased (value of the coin). The function does not allow you to burn more coins than are available in the supply.

The deny list is only applicable to regulated coins. As mentioned previously, when you create a regulated coin you receive a DenyCapV2 that authorizes the bearer to add and remove addresses from the system-created DenyList object. Any address on the list for your coin is unable to use the coin as an input to transactions starting immediately upon being added. At the epoch that follows address addition to the deny list, the addresses additionally cannot receive the coin type. In other words, an address that gets added to the deny list for a coin type is immediately unable to send the coin. At the start of the following epoch, the address is still unable to send the coin but is also unable to receive it. From that point, the address cannot interact with the coin until expressly removed from the deny list by the DenyCapV2 bearer.

Use the coin::deny_list_v2_add function to add the provided address to the deny list for your coin. The signature for the function is:

When using this function, you provide the DenyList object (0x403), the DenyCap you receive on coin creation, the address to add to the list, and the transaction context. After using this function, the address you provide is unable to use your coin by the next epoch.

Use the coin::deny_list_v2_remove function to remove addresses from the deny list for your coin.

When using this function, you provide the DenyList object (0x403), the DenyCapV2 you receive on coin creation, the address to remove from the list, and the transaction context. If you try to remove an address that isn't on the list, you receive an ENotFrozen error and the function aborts. After calling this function, the address you provide is able to use your coin by the next epoch.

You can use either the TypeScript or Rust SDK to manipulate the addresses held in the DenyList for your coin. The following examples are based on the regulated coin sample .

Globally pausing coin activity is only applicable to regulated coin types.

To pause activity across the network for a regulated coin type with the allow_global_pause field set to true , use coin::deny_list_v2_enable_global_pause . You must provide the DenyCapV2 object for the coin type to initiate the pause.

Transaction activity is paused immediately, and no addresses can receive the coin in the epoch that follows the call to pause.

To restart network activity for a paused regulated coin, use the coin::deny_list_v2_disable_global_pause function. As with pausing, you must provide the DenyCapV2 object for the coin type. Transaction activity resumes immediately, and addresses can begin receiving the coin in the epoch that follows the call to remove the pause.

You can use the following functions to retrieve data from coins.

Use the following functions to get the values for the respective fields on the metadata object for coins.

Use the coin::supply function to get the current supply of a given coin.

If the CoinMetadata object was not frozen upon creation, you can use the following functions to update its values.

Each function signature is similar. Replace and with the values defined in the table to get the signature of each function:

RegulatedCoinMetadata is frozen upon creation, so there are no functions to update its data.

Check out the following content for more information about coins and tokens on Sui:

Regulated coins

The Coin standard includes the ability to create regulated coins. To create a regulated coin, you use the coin::create_regulated_currency_v2 function (which uses the coin::create_currency function itself), but which also returns a DenyCap capability. The DenyCap capability allows the bearer to maintain a list of addresses that aren't allowed to use the token.

The regulated-coin-sample repository provides an example of regulated coin creation.

The list of addresses that aren't able to use a particular regulated coin is held within a system-created DenyList shared object. If you have access to the DenyCap , then you can use the coin::deny_list_v2_add and coin::deny_list_v2_remove functions to add and remove addresses.

Regulated coin objects include an allow_global_pause Boolean field. When set to true , the bearer of the DenyCapV2 object for the coin type can use the coin::deny_list_v2_enable_global_pause function to pause coin activity indefinitely. Immediately upon the bearer initiating the pause, the network disallows the coin type as input for any transactions. At the start of the next epoch (epochs last ~24 hours), the network additionally disallows all addresses from receiving the coin type.

When the bearer of the DenyCapV2 object for the coin type removes the pause using coin::deny_list_v2_disable_global_pause , the coins are immediately available to use again as transaction inputs. Addresses cannot receive the coin type, however, until the following epoch.

The global pause functionality does not affect the deny list for the coin. After clearing the pause for the coin, any addresses included in the deny list are still unable to interact with the coin.

Each coin you create includes metadata that describes it. Typically, smart contracts freeze this object upon creation using the transfer::public_freeze_object function because the metadata for coins should almost never change. Regulated coins freeze the metadata they create automatically.

Regular coins using the Coin standard include a CoinMetadata object. As mentioned previously, regulated coins build on top of the same procedure that creates regular coins, so they receive the same metadata object in addition to a RegulatedCoinMetadata object that includes deny list information.

The fields of the metadata objects include the following:

The coin module provides the logic for creating and destroying coins on the Sui network (as long as you own the associated TreasuryCap). These functions are the same for all coins and each requires the TreasuryCap as an input.

Use the coin::mint function to create new tokens.

The signature shows that a Coin results from calling the function with a TreasuryCap , value for the coin created, and the transaction context. The function updates the total supply in TreasuryCap automatically. Upon display, the coin value respects the decimals value in the metadata. So, if you supply 1000000 as the coin value that has a decimal value of 6 , the coin's value displays as 1.000000 .

Use the coin::burn function to destroy current tokens.

The signature shows that only the TreasuryCap and coin object you want to burn are necessary inputs, returning the amount by which the supply was decreased (value of the coin). The function does not allow you to burn more coins than are available in the supply.

The deny list is only applicable to regulated coins. As mentioned previously, when you create a regulated coin you receive a DenyCapV2 that authorizes the bearer to add and remove addresses from the system-created DenyList object. Any address on the list for your coin is unable to use the coin as an input to transactions starting immediately upon being added. At the epoch that follows address addition to the deny list, the addresses additionally cannot receive the coin type. In other words, an address that gets added to the deny list for a coin type is immediately unable to send the coin. At the start of the following epoch, the address is still unable to send the coin but is also unable to receive it. From that point, the address cannot interact with the coin until expressly removed from the deny list by the DenyCapV2 bearer.

Use the coin::deny_list_v2_add function to add the provided address to the deny list for your coin. The signature for the function is:

When using this function, you provide the DenyList object (0x403), the DenyCap you receive on coin creation, the address to add to the list, and the transaction context. After using this function, the address you provide is unable to use your coin by the next epoch.

Use the coin::deny_list_v2_remove function to remove addresses from the deny list for your coin.

When using this function, you provide the DenyList object (0x403), the DenyCapV2 you receive on coin creation, the address to remove from the list, and the transaction context. If you try to remove an address that isn't on the list, you receive an ENotFrozen error and the function aborts. After calling this function, the address you provide is able to use your coin by the next epoch.

You can use either the TypeScript or Rust SDK to manipulate the addresses held in the DenyList for your coin. The following examples are based on the regulated coin sample .

Globally pausing coin activity is only applicable to regulated coin types.

To pause activity across the network for a regulated coin type with the allow_global_pause field set to true , use coin::deny_list_v2_enable_global_pause . You must provide the DenyCapV2 object for the coin type to initiate the pause. Transaction activity is paused immediately, and no addresses can receive the coin in the epoch that follows the call to pause.

To restart network activity for a paused regulated coin, use the coin::deny_list_v2_disable_global_pause function. As with pausing, you must provide the DenyCapV2 object for the coin type. Transaction activity resumes immediately, and addresses can begin receiving the coin in the epoch that follows the call to remove the pause.

You can use the following functions to retrieve data from coins.

Use the following functions to get the values for the respective fields on the metadata object for coins.

Use the coin::supply function to get the current supply of a given coin.

If the CoinMetadata object was not frozen upon creation, you can use the following functions to update its values.

Each function signature is similar. Replace and with the values defined in the table to get the signature of each function:

RegulatedCoinMetadata is frozen upon creation, so there are no functions to update its data.

Check out the following content for more information about coins and tokens on Sui:

Coin metadata

Each coin you create includes metadata that describes it. Typically, smart contracts freeze this object upon creation using the transfer::public_freeze_object function because the metadata for coins should almost never change. Regulated coins freeze the metadata they create automatically.

Regular coins using the Coin standard include a CoinMetadata object. As mentioned previously, regulated coins build on top of the same procedure that creates regular coins, so they receive the same metadata object in addition to a RegulatedCoinMetadata object that includes deny list information.

The fields of the metadata objects include the following:

The coin module provides the logic for creating and destroying coins on the Sui network (as long as you own the associated TreasuryCap). These functions are the same for all coins and each requires the TreasuryCap as an input.

Use the coin::mint function to create new tokens.

The signature shows that a Coin results from calling the function with a TreasuryCap , value for the coin created, and the transaction context. The function updates the total supply in TreasuryCap automatically. Upon display, the coin value respects the decimals value in the metadata. So, if you supply 1000000 as the coin value that has a decimal value of 6 , the coin's value displays as 1.000000 .

Use the coin::burn function to destroy current tokens.

The signature shows that only the TreasuryCap and coin object you want to burn are necessary inputs, returning the amount by which the supply was decreased (value of the coin). The function does not allow you to burn more coins than are available in the supply.

The deny list is only applicable to regulated coins. As mentioned previously, when you create a regulated coin you receive a DenyCapV2 that authorizes the bearer to add and remove addresses from the system-created DenyList object. Any address on the list for your coin is unable to use the coin as an input to transactions starting immediately upon being added. At the epoch that follows address addition to the deny list, the addresses additionally cannot receive the coin type. In other words, an address that gets added to the deny list for a coin type is immediately unable to send the coin. At the start of the following epoch, the address is still unable to send the coin but is also unable to receive it. From that point, the address cannot interact with the coin until expressly removed from the deny list by the DenyCapV2 bearer.

Use the coin::deny_list_v2_add function to add the provided address to the deny list for your coin. The signature for the function is:

When using this function, you provide the DenyList object (0x403), the DenyCap you receive on coin creation, the address to add to the list, and the transaction context. After using this function, the address you provide is unable to use your coin by the next epoch.

Use the coin::deny_list_v2_remove function to remove addresses from the deny list for your coin.

When using this function, you provide the DenyList object (0x403), the DenyCapV2 you receive on coin creation, the address to remove from the list, and the transaction context. If you try to remove an address that isn't on the list, you receive an ENotFrozen error and the function aborts. After calling this function, the address you provide is able to use your coin by the next epoch.

You can use either the TypeScript or Rust SDK to manipulate the addresses held in the DenyList for your coin. The following examples are based on the regulated coin sample .

Globally pausing coin activity is only applicable to regulated coin types.

To pause activity across the network for a regulated coin type with the allow_global_pause field set to true , use coin::deny_list_v2_enable_global_pause . You must provide the DenyCapV2 object for the coin type to initiate the pause. Transaction activity is paused immediately, and no addresses can receive the coin in the epoch that follows the call to pause.

To restart network activity for a paused regulated coin, use the coin::deny_list_v2_disable_global_pause function. As with pausing, you must provide the DenyCapV2 object for the coin type. Transaction activity resumes immediately, and addresses can begin receiving the coin in the epoch that follows the call to remove the pause.

You can use the following functions to retrieve data from coins.

Use the following functions to get the values for the respective fields on the metadata object for coins.

Use the coin::supply function to get the current supply of a given coin.

If the CoinMetadata object was not frozen upon creation, you can use the following functions to update its values.

Each function signature is similar. Replace and with the values defined in the table to get the signature of each function:

RegulatedCoinMetadata is frozen upon creation, so there are no functions to update its data.

Check out the following content for more information about coins and tokens on Sui:

Minting and burning coins

The coin module provides the logic for creating and destroying coins on the Sui network (as long as you own the associated TreasuryCap). These functions are the same for all coins and each requires the TreasuryCap as an input.

Use the coin::mint function to create new tokens.

The signature shows that a Coin results from calling the function with a TreasuryCap , value for the coin created, and the transaction context. The function updates the total supply in TreasuryCap automatically. Upon display, the coin value respects the decimals value in the metadata. So, if you supply 1000000 as the coin value that has a decimal value of 6 , the coin's value displays as 1.000000 .

Use the coin::burn function to destroy current tokens.

The signature shows that only the TreasuryCap and coin object you want to burn are necessary inputs, returning the amount by which the supply was decreased (value of the coin). The function does not allow you to burn more coins than are available in the supply.

The deny list is only applicable to regulated coins. As mentioned previously, when you create a regulated coin you receive a DenyCapV2 that authorizes the bearer to add and remove addresses from the system-created DenyList object. Any address on the list for your coin is unable to use the coin as an input to transactions starting immediately upon being added. At the epoch that follows address addition to the deny list, the addresses additionally cannot receive the coin type. In other words, an address that gets added to the deny list for a coin type is immediately unable to send the coin. At the start of the following epoch, the address is still unable to send the coin but is also unable to receive it. From that point, the address cannot interact with the coin until expressly removed from the deny list by the DenyCapV2 bearer.

Use the coin::deny_list_v2_add function to add the provided address to the deny list for your coin. The signature for the function is:

When using this function, you provide the DenyList object (0x403), the DenyCap you receive on coin creation, the address to add to the list, and the transaction context. After using this function, the address you provide is unable to use your coin by the next epoch.

Use the coin::deny_list_v2_remove function to remove addresses from the deny list for your coin.

When using this function, you provide the DenyList object (0x403), the DenyCapV2 you receive on coin creation, the address to remove from the list, and the transaction context. If you try to remove an address that isn't on the list, you receive an ENotFrozen error and the function aborts. After calling this function, the address you provide is able to use your coin by the next epoch.

You can use either the TypeScript or Rust SDK to manipulate the addresses held in the DenyList for your coin. The following examples are based on the regulated coin sample .

Globally pausing coin activity is only applicable to regulated coin types.

To pause activity across the network for a regulated coin type with the allow_global_pause field set to true , use coin::deny_list_v2_enable_global_pause . You must provide the DenyCapV2 object for the coin type to initiate the pause. Transaction activity is paused immediately, and no addresses can receive the coin in the epoch that follows the call to pause.

To restart network activity for a paused regulated coin, use the coin::deny_list_v2_disable_global_pause function. As with pausing, you must provide the DenyCapV2 object for the coin type. Transaction activity resumes immediately, and addresses can begin receiving the coin in the epoch that follows the call to remove the pause.

You can use the following functions to retrieve data from coins.

Use the following functions to get the values for the respective fields on the metadata object for coins.

Use the coin::supply function to get the current supply of a given coin.

If the CoinMetadata object was not frozen upon creation, you can use the following functions to update its values.

Each function signature is similar. Replace and with the values defined in the table to get the signature of each function:

RegulatedCoinMetadata is frozen upon creation, so there are no functions to update its data.

Check out the following content for more information about coins and tokens on Sui:

Adding and removing addresses to and from the deny list

The deny list is only applicable to regulated coins. As mentioned previously, when you create a regulated coin you receive a DenyCapV2 that authorizes the bearer to add and remove addresses from the system-created DenyList object. Any address on the list for your coin is unable to use the coin as an input to transactions starting immediately upon being added. At the epoch that follows address addition to the deny list, the addresses additionally cannot receive the coin type. In other words, an address that gets added to the deny list for a coin type is immediately unable to send the coin. At the start of the following epoch, the address is still unable to send the coin but is also unable to receive it. From that point, the address cannot interact with the coin until expressly removed from the deny list by the DenyCapV2 bearer.

Use the coin::deny_list_v2_add function to add the provided address to the deny list for your coin. The signature for the function is:

When using this function, you provide the DenyList object (0x403), the DenyCap you receive on coin creation, the address to add to the list, and the transaction context. After using this function, the address you provide is unable to use your coin by the next epoch.

Use the coin::deny_list_v2_remove function to remove addresses from the deny list for your coin.

When using this function, you provide the DenyList object (0x403), the DenyCapV2 you receive on coin creation, the address to remove from the list, and the transaction context. If you try to remove an address that isn't on the list, you receive an ENotFrozen error and the function aborts. After calling this function, the address you provide is able to use your coin by the next epoch.

You can use either the TypeScript or Rust SDK to manipulate the addresses held in the DenyList for your coin. The following examples are based on the regulated coin sample .

Globally pausing coin activity is only applicable to regulated coin types.

To pause activity across the network for a regulated coin type with the allow_global_pause field set to true , use coin::deny_list_v2_enable_global_pause . You must provide the DenyCapV2 object for the coin type to initiate the pause. Transaction activity is paused immediately, and no addresses can receive the coin in the epoch that follows the call to pause.

To restart network activity for a paused regulated coin, use the coin::deny_list_v2_disable_global_pause function. As with pausing, you must provide the DenyCapV2 object for the coin type. Transaction activity resumes immediately, and addresses can begin receiving the coin in the epoch that follows the call to remove the pause.

You can use the following functions to retrieve data from coins.

Use the following functions to get the values for the respective fields on the metadata object for coins.

Use the coin::supply function to get the current supply of a given coin.

If the CoinMetadata object was not frozen upon creation, you can use the following functions to update its values.

Each function signature is similar. Replace and with the values defined in the table to get the signature of each function:

RegulatedCoinMetadata is frozen upon creation, so there are no functions to update its data.

Check out the following content for more information about coins and tokens on Sui:

Globally pausing and unpausing regulated coin activity

Globally pausing coin activity is only applicable to regulated coin types.

To pause activity across the network for a regulated coin type with the allow_global_pause field set to true , use coin::deny_list_v2_enable_global_pause . You must provide the DenyCapV2 object for the coin type to initiate the pause. Transaction activity is paused immediately, and no addresses can receive the coin in the epoch that follows the call to pause.

To restart network activity for a paused regulated coin, use the coin::deny_list_v2_disable_global_pause function. As with pausing, you must provide the DenyCapV2 object for the coin type. Transaction activity resumes immediately, and addresses can begin receiving the coin in the epoch that follows the call to remove the pause.

You can use the following functions to retrieve data from coins.

Use the following functions to get the values for the respective fields on the metadata object for coins.

Use the coin::supply function to get the current supply of a given coin.

If the CoinMetadata object was not frozen upon creation, you can use the following functions to update its values.

Each function signature is similar. Replace and with the values defined in the table to get the signature of each function:

RegulatedCoinMetadata is frozen upon creation, so there are no functions to update its data.

Check out the following content for more information about coins and tokens on Sui:

Query coin data

You can use the following functions to retrieve data from coins.

Use the following functions to get the values for the respective fields on the metadata object for coins.

Use the coin::supply function to get the current supply of a given coin.

If the CoinMetadata object was not frozen upon creation, you can use the following functions to update its values.

Each function signature is similar. Replace and with the values defined in the table to get the signature of each function:

RegulatedCoinMetadata is frozen upon creation, so there are no functions to update its data.

Check out the following content for more information about coins and tokens on Sui:

Update coin metadata

If the CoinMetadata object was not frozen upon creation, you can use the following functions to update its values.

Each function signature is similar. Replace and with the values defined in the table to get the signature of each function:

RegulatedCoinMetadata is frozen upon creation, so there are no functions to update its data.

Check out the following content for more information about coins and tokens on Sui:

Related links

Check out the following content for more information about coins and tokens on Sui: