Snaps API

Snaps can communicate with and modify the functionality of MetaMask using the Snaps API . To call each method, you must first request permission in the Snap manifest file.

snap_dialog

Displays a dialog in the MetaMask UI. There are three types of dialogs with different parameters and return types alert ,confirmation , and prompt .

caution Dialogs do not work when MetaMask is locked. To check if MetaMask is locked, use snap_getClientStatus .

Alert dialog

Displays an alert that can only be acknowledged.

Parameters

An object containing the contents of the alert dialog:

- type
-
  - The type of dialog ("alert"
-).
- One of:* content
-
  -
    - The content of the alert, as a custom UI
-
  - component.
-
  - id
-
  -
    - The ID of an interactive interface
-
  - .

Example

index.js import

{ panel , text , heading }

from

"@metamask/snaps-sdk" ;

await snap . request ({ method :

"snap_dialog" , params :

{ type :

"alert" , content :

panel ([heading ("Something happened in the system") , text ("The thing that happened is...") ,]) , } , }) ;

// Code that should execute after the alert has been acknowledged.

Confirmation dialog

Displays a confirmation that can be accepted or rejected.

Parameters

An object containing the contents of the confirmation dialog:

- type
-
    - The type of dialog ("confirmation"
-).
- One of:* content
-
    -
        - The content of the confirmation, as acustom UI
-
    - component.
-
    - id
-
    -
        - The ID of aninteractive interface
-
    - .

Returns

true if the confirmation was accepted,false otherwise.

Example

index.js import

{ panel , text , heading }

from

"@metamask/snaps-sdk" ;

const result =

await snap . request ({ method :

"snap_dialog" , params :

{ type :

"confirmation" , content :

panel ([heading ("Would you like to take the action?") , text ("The action is...") ,]) , } , }) ;

if

(result ===

true)

{ // Do the action. }

Prompt dialog

Displays a prompt where the user can enter a text response.

Parameters

An object containing the contents of the prompt dialog:

- type
-
    - The type of dialog ("prompt"
-).
- placeholder
-
    - Text that will be in the input field when nothing is typed.
- One of:* content

-
  -
    - The content of the confirmation, as acustom UI
-
  - component.
-
  - id
-
  -
    - The ID of aninteractive interface
-
  - .

Returns

The text entered by the user if the prompt was submitted ornull if the prompt was rejected or closed. If the user does not enter any text and submits the prompt, the value is an empty string.

Example

index.js import

{ panel , text , heading }

from

"@metamask/snaps-sdk" ;

const walletAddress =

await snap . request ({ method :

"snap_dialog" , params :

{ type :

"prompt" , content :

panel ([heading ("What is the wallet address?") , text ("Please enter the wallet address to be monitored") ,]) , placeholder :

"0x123..." , } , }) ;

// walletAddress will be a string containing the address entered by the user.

snap_getBip32Entropy

important If you call this method, you receive the user's parent key for the derivation path they request. You're managing the user's keys and assets on their behalf. Their safety is your responsibility. Gets theSLIP-10 key for thepath andcurve specified by the method name.

This method is designed to be used with the@metamask/key-tree module.@metamask/key-tree can help you get theextended private keys for user addresses, but it's your responsibility to know how to use those keys to, for example, derive an address for the relevant protocol or sign a transaction for the user.

Parameters

An object containing:

- path
-
  - An array, starting withm
- , containing the BIP-32 derivation path to the key to
- retrieve.
- For example,["m", "44'", "60'"]
- .
- curve

-
    - The curve to use for the key derivation.
- Must be"ed25519"
- or"secp256k1"
- .

Returns

An object representing theSLIP-10 HD tree node and containing its corresponding key material:

- depth
-
    - The 0-indexed path depth of the node.
- parentFingerprint
-
    - The fingerprint of the parent key, or 0 if this is a master node.
- index
-
    - The index of the node, or 0 if this is a master node.
- privateKey
-
    - The private key of the node.
- publicKey
-
    - The public key of the node.
- chainCode
-
    - The chain code of the node.
- curve
-
    - The name of the curve used by the node:"ed25519"
- or"secp256k1"
- .

Example

- Manifest file
- JavaScript

snap.manifest.json "initialPermissions" :

{ "snap_getBip32Entropy" :

[{ "path" :

["m" ,

"44'" ,

"3'"] , "curve" :

"secp256k1"

// Or "ed25519" }] } index.js import

{

SLIP10Node

}

from

"@metamask/key-tree" ;

// This example uses Dogecoin, which has a derivation path starting with m/44'/3'. const dogecoinNode =

await snap . request ({ method :

"snap_getBip32Entropy" , params :

{ // The path and curve must be specified in the initial permissions. path :

["m" ,

"44'" ,

"3'"] , curve :

"secp256k1" , } , }) ;

// Next, create an instance of a SLIP-10 node for the Dogecoin node. const dogecoinSlip10Node =

await

SLIP10Node . fromJSON (dogecoinNode) ;

// m/44'/3'/0' const accountKey0 =

await dogecoinSlip10Node . derive (["bip32:0'"]) ;

// m/44'/3'/1' const accountKey1 =

await dogecoinSlip10Node . derive (["bip32:1'"]) ;

// Now, you can ask the user to sign transactions, etc.

snap_getBip32PublicKey

Gets theBIP-32 public key for the derivation path specified by thepath parameter. Note that this returns the public key, not the extended public key (xpub), or Ethereum address.

Parameters

An object containing:

- path
-
    - An array, starting withm
- , containing the BIP-32 derivation path to the public key to
- retrieve.
- For example,["m", "44'", "60'"]
- .
- curve
-
    - The curve to use for the key derivation.
- Must be"ed25519"
- or"secp256k1"
- .
- compressed
-
    - Indicates whether the public key should be compressed.
- The default isfalse
- .

Returns

The public key as hexadecimal string.

Example

- Manifest file
- JavaScript

snap.manifest.json "initialPermissions" :

{ "snap_getBip32PublicKey" :

[{ "path" :

["m" ,

"44'" ,

"3'" ,

"0'" ,

"0" ,

"0"] , "curve" :

"secp256k1" }] } index.js // This example uses Dogecoin, which has a derivation path starting with m/44'/3'. const dogecoinPublicKey =

await snap . request ({ method :

"snap_getBip32PublicKey" , params :

{ // The path and curve must be specified in the initial permissions. path :

["m" ,

"44'" ,

"3'" ,

"0'" ,

"0" ,

"0"] , curve :

"secp256k1" , compressed :

false , } , }) ;

// "0x..." console . log (dogecoinPublicKey) ;

snap_getBip44Entropy

important If you call this method, you receive the user's parent key for the protocol they request. You're managing the user's keys and assets on their behalf. Their safety is your responsibility. Gets theBIP-44 coin_type key for thecoin_type number specified by the method name. SeeSLIP-44 for the list of available protocols and theircoin_type values.

This method is designed to be used with the@metamask/key-tree module.@metamask/key-tree can help you get theextended private keys for user addresses, but it's your responsibility to know how to use those keys to, for example, derive an address for the relevant protocol or sign a transaction for the user.

Parameters

An object containingcoinType , the BIP-44 coin type to get the entropy for.

caution Coin type 60 is reserved for MetaMask externally owned accounts and blocked for Snaps. If you wish to connect to MetaMask accounts in a Snap, useendowment:ethereum-provider andeth_requestAccounts .

Returns

An object representing theBIP-44 coin_type HD tree node and containing its corresponding key material:

- coin_type
-
    - The BIP-44coin type
- value of the node.
- depth
-
    - The 0-indexed BIP-44 path depth of the node.
- Since this is acoin_type
- node, the depth is2

- .
- privateKey
-
    - The hexadecimal-encoded string representation of the private key for the node.
- publicKey
-
    - The hexadecimal-encoded string representation of the public key for the node.
- chainCode
-
    - The hexadecimal-encoded string representation of the chain code for the node.
- path
-
    - A human-readable representation of the BIP-44 HD tree path of the node.
- Since this is acoin_type
- node, the path is of the formm/44'/coin_type'
- .

Example

- Manifest file
- JavaScript

snap.manifest.json "initialPermissions" :

{ "snap_getBip44Entropy" :

[{ "coinType" :

3 }] } index.js import

{ getBIP44AddressKeyDeriver }

from

"@metamask/key-tree" ;

// This example uses Dogecoin, which has coin_type 3. const dogecoinNode =

await snap . request ({ method :

"snap_getBip44Entropy" , params :

{ coinType :

3 , } , }) ;

// Next, create an address key deriver function for the Dogecoin coin_type node. // In this case, its path is: m/44'/3'/0'/0/address_index const deriveDogecoinAddress =

await

getBIP44AddressKeyDeriver (dogecoinNode) ;

// These are BIP-44 nodes containing the extended private keys for the respective derivation paths.

// m/44'/3'/0'/0/0 const addressKey0 =

await

deriveDogecoinAddress (0) ;

// m/44'/3'/0'/0/1 const addressKey1 =

await

deriveDogecoinAddress (1) ;

// Now, you can ask the user to sign transactions, etc.

snap_getClientStatus

Gets the locked status of the Snaps client.

It is useful to check if MetaMask is locked in the following situations:

- When running background operations that require MetaMask to be unlocked, for example[accessing encrypted state]
- . If MetaMask is locked, the user gets a popup asking them to enter their password, which might be unexpected or confusing.
- When[displaying a dialog]
- . Dialogs do not work when MetaMask is locked.

Returns

true if MetaMask is locked,false if MetaMask is unlocked.

Example

index.js import

type

{ OnCronjobHandler }

from

"@metamask/snaps-sdk" ; import

{ MethodNotFoundError }

from

"@metamask/snaps-sdk" ;

export

const onCronjob :

OnCronjobHandler

=

async

({ request })

=>

{ switch

(request . method)

{ case

"execute" : // Find out if MetaMask is locked. const

{ locked }

=

await snap . request ({ method :

"snap_getClientStatus" }) ;

if

(! locked)

{ // Do something that requires MetaMask to be unlocked, such as access encrypted state. }

default : throw

new

MethodNotFoundError ( ) ; } } ;

snap_getEntropy

Gets a deterministic 256-bit entropy value, specific to the Snap and the user's account. You can use this entropy to generate a private key, or any other value that requires a high level of randomness. Other Snaps can't access this entropy, and it changes if the user's secret recovery phrase changes.

You can optionally specify a salt to generate different entropy for different purposes. Using a salt results in entropy unrelated to the entropy generated without a salt.

This value is deterministic: it's always the same for the same Snap, user account, and salt.

Parameters

An object containing:

- version
-
    - The number1
- .
- This is reserved for future use.
- salt
- (optional) - An arbitrary string to be used as a salt for the entropy.
- This can be used to generate different entropy for different purposes.

Returns

The entropy as a hexadecimal string.

Example

- Manifest file
- JavaScript

snap.manifest.json "initialPermissions" :

{ "snap_getEntropy" :

{ } } index.js const entropy =

await snap . request ({ method :

"snap_getEntropy" , params :

{ version :

1 , salt :

"foo" ,

// Optional. } , }) ;

// "0x..." console . log (entropy) ;

snap_getFile

Gets a static file's content in UTF-8, Base64, or hexadecimal. The file must be specified in the Snap's manifest file.

Parameters

An object containing:

- path
-

    - The path to the file, relative to the Snap's package directory (that is, one level abovesrc
-).
- encoding
- (optional) - One ofutf8
- ,base64
- , orhex
- . The default isbase64
- .

Returns

The file content as a string in the requested encoding.

Example

- Manifest file
- JavaScript

snap.manifest.json "source" :

{ "shasum" :

"xxx" , "location" :

{ // ... } , "files" :

["./files/myfile.bin"] } index.js const contents =

await snap . request ({ method :

"snap_getFile" , params :

{ path :

"./files/myfile.bin" , encoding :

"hex" , } , }) ;

// "0x..." console . log (contents) ;

snap_getLocale

Gets the user's locale setting. You can use this method to localize text in your snap.

Returns

The user's locale setting as alanguage code .

Example

index.js import

{ panel , text }

from

"@metamask/snaps-sdk" ;

const locale =

await snap . request ({

method :

"snap_getLocale"

}) ;

let greeting =

"Hello" ; if (locale ===

"es")

{ greeting =

"Hola" ; }

await snap . request ({ method :

"snap_dialog" , params :

{ type :

"alert" , content :

panel ([text (greeting) ,]) , } , }) ;

snap_manageAccounts

Managesaccount management Snap accounts. This method is organized into multiple sub-methods which each take their own parameters:

- createAccount
- updateAccount
- deleteAccount
- listAccounts
- submitResponse

createAccount

Creates a new Snap account.

note The Snap is responsible for maintaining its own record of accounts. This can be done usingsnap_manageState .

Parameters

account -An account object.

Returns

null

Example

index.ts import

{ Keyring , KeyringAccount }

from

"@metamask/keyring-api" ;

class

MyKeyring

implements

Keyring

{ // Other methods.

async

createAccount (name :

```
string , options : Record < string , Json

|

null

=

null , ) :

Promise < KeyringAccount

{

const account : KeyringAccount =

{ id :

uuid ( ) , name , options , address , supportedMethods :

[ "eth_sendTransaction" , "eth_sign" , "eth_signTransaction" , "eth_signTypedData_v1" , "eth_signTypedData_v2" ,
"eth_signTypedData_v3" , "eth_signTypedData_v4" , "eth_signTypedData" , "personal_sign" , ] , type :

"eip155:eoa" , } ;

// Store the account in state.

await snap . request ( { method :

"snap_manageAccounts" , params :

{ method :

"createAccount" , params :

{ account } , } , } ) ;

return account ; } }
```

updateAccount

Updates an existing Snap account.

note The Snap is responsible for maintaining its own record of accounts. This can be done using snap_manageState .

Parameters

account - An account object.

Returns

null

Example

```
index.ts import

{ Keyring , KeyringAccount }

from

"@metamask/keyring-api" ;

class

MyKeyring

implements

Keyring

{ // Other methods.
```

async

updateAccount (account : KeyringAccount) :

Promise < void

{ // Store the new account details in state.

await snap . request ({ method :

"snap_manageAccounts" , params :

{ method :

"updateAccount" , params :

{ account } , } , }) ; } }

deleteAccount

Deletes a Snap account.

note The Snap is responsible for maintaining its own record of accounts. This can be done using snap_manageState .

Parameters

id - The ID of the account to be deleted.

Returns

null

Example

index.ts import

{ Keyring }

from

"@metamask/keyring-api" ;

class

MyKeyring

implements

Keyring

{ // Other methods.

async

deleteAccount (id :

string) :

Promise < void

{ // Delete the account from state.

await snap . request ({ method :

"snap_manageAccounts" , params :

{ method :

"deleteAccount" , params :

{ id } , } , }) ; } }

listAccounts

Lists the calling Snap's accounts that are known to MetaMask. This method does not call back to the Snap. Instead, the Snap can use it to check whether there's a discrepancy between the Snap's internal state of accounts and the state known to MetaMask.

Returns

An array of[account objects](#) .

Example

index.ts import

{ Keyring , KeyringAccount }

from

"@metamask/keyring-api" ;

class

MyKeyring

implements

Keyring

{ // Other methods.

async

checkIfAccountsInSync ( ) :

Promise < boolean

{

const knownAccounts : KeyringAccount [ ]

=

/ *Grab accounts from Snap state.* / ;

const listedAccounts : KeyringAccount [ ]

=

await snap . request ({ method :

"snap_manageAccounts" , params :

{ method :

"listAccounts" , } , }) ;

// Compare the arrays and return the response. } }

submitResponse

Finalizes a signing request. This is usually called as part of the[keyring_approveRequest](#) method.

Parameters

- id
-
    - The ID of the request to finalize.
- result
-
    - The result that should be returned to the original JSON-RPC caller.

Returns

null

Example

index.ts import

{ Keyring }

from

"@metamask/keyring-api" ; import

{ Json }

from

"@metamask/utils" ;

class

MyKeyring

implements

Keyring

{ // Other methods.

async

approveRequest (id :

string , result ? : Json) :

Promise < void

{ // Do any Snap-side logic to finish approving the request.

await snap . request ({ method :

"snap_manageAccounts" , params :

{ method :

"submitResponse" , params :

{ id , result } , } , }) ; } }

snap_manageState

Allows the Snap to persist up to 100 MB of data to disk and retrieve it at will. By default, the data is automatically encrypted using a Snap-specific key and automatically decrypted when retrieved. You can setencrypted tofalse to use unencrypted storage.

note Accessing encrypted state requires MetaMask to be unlocked. If you need to access encrypted state in a background task such as a cron job, you can usesnap_getClientStatus to ensure that MetaMask is unlocked, preventing an unexpected password request popup.

Parameters

An object containing:

- operation
-
    - The state operation to perform ("clear"
- ,"get"
- , or"update"

-).
- newState
-
    - The value to update state with if the operation is"update"
- , and nothing otherwise.
- encrypted
- (optional) - Indicates whether the Snap will encrypt the data.
- The default istrue
- .
- If set tofalse
- , the Snap will use a separate storage section, and will not encrypt the data.
- This is useful to access the data from background operations without requiring the user to enter
- their password in the case that MetaMask is locked.

Returns

The value stored in state if the operation isget , andnull otherwise.

Example

index.js // Persist some data. await snap . request ({ method :

"snap_manageState" , params :

{ operation :

"update" , newState :

{

hello :

"world"

} , } , }) ;

// At a later time, get the data stored. const persistedData =

await snap . request ({ method :

"snap_manageState" , params :

{

operation :

"get"

} , }) ;

console . log (persistedData) ; // { hello: "world" }

// If there's no need to store data anymore, clear it out. await snap . request ({ method :

"snap_manageState" , params :

{ operation :

"clear" , } , }) ;

snap_notify

Displays a notification in MetaMask or natively in the browser. Snaps can trigger a short notification text for actionable or time sensitive information.

Notification rate limits The ability for Snaps to trigger notifications is rate-limited to:

- 2 native notifications per 5 minutes per Snap.
- 5 in-app notifications per minute per Snap.

Parameters

An object containing the contents of the notification:

- type
-
    - The notification type.
- SpecifyinApp
- to display the notification in the MetaMask UI, andnative
- to display the
- notification in the browser.
- We recommend usinginApp
- because there's no guarantee that native notifications are displayed to
- the user.
- message
-
    - A message to show in the notification.

Example

index.js await snap . request ({ method :

"snap_notify" , params :

{ type :

"inApp" , message :

"Hello, world!" , } , }) ;

Interactive UI methods

The following methods are used ininteractive UI . These methods do not require requesting permission in the Snap manifest file.

snap_createInterface

Flask Only This feature is experimental and only available inMetaMask Flask , the canary distribution of MetaMask. Creates an interactive interface for use ininteractive UI .

Parameters

An object containing:

- ui
-
    - Thecustom UI
- to create.

Returns

The interface's ID to be used insnap_dialog , returned fromonTransaction oronHomePage .

Example

index.js const interfaceId =

await snap . request ({ method :

"snap_createInterface" , params :

{ ui :

panel ([heading ("Interactive interface") , button ({ value :

"Click me" , name :

"interactive-button" , }) ,]) } , }) ;

await snap . request ({ method :

"snap_dialog" , params :

{ type :

"Alert" , id : interfaceId } }) ;

snap_getInterfaceState

Flask Only This feature is experimental and only available inMetaMask Flask , the canary distribution of MetaMask. Gets the state of an interactive interface by its ID. For use ininteractive UI .

Parameters

- id
-
    - The ID of the interface.

Returns

An object where each top-level property can be one of the following:

- Thename
- of aninput
- with its current value.
- Thename
- of aform
- , with a nested object containing the
- current values of allinputs
- in the form.

Example

index.js const interfaceId =

await snap . request ({ method :

"snap_createInterface" , params :

{ ui :

panel ([heading ("Interactive UI Example Snap") , // A top-level input. input ({ name :

"top-level-input" , placeholder :

"Enter something" , }) , // A top-level form... form ({ name :

"example-form" , children :

[// ...with a nested input. input ({ name :

"nested-input" , placeholder :

"Enter something" , }) , button ("Submit" ,

ButtonType . Submit ,

"submit") ,] , }) ,]) , } , }) ;

const state =

await snap . request ({ method :

"snap_getInterfaceState" , params :

{ id : interfaceId , } , }) ;

console . log (state) ; /*{ "top-level-input": "What the user typed in that field", "example-form": { "nested-input": "What the user typed in that field" } } */*

snap_updateInterface

Flask Only This feature is experimental and only available inMetaMask Flask , the canary distribution of MetaMask. Updates an interactive interface. For use ininteractive UI .

Parameters

An object containing:

- id
-
  - The ID of the interface to be updated, usually received in theonUserInput
- entry point.
- ui
-
  - Thecustom UI
- to create.

Example

index.js export

function

onUserInput ({ id , event })

{ console . log ("Received input event" , event) ;

await snap . request ({ method :

"snap_updateInterface" , params :

{ id , ui :

panel ([heading ("New interface") ,]) , } , }) ; } ;

Edit this page