Snaps entry points

Snaps can expose the following entry points.

onCronjob

To runcron jobs for the user, a Snap must expose theonCronjob entry point. MetaMask calls theonCronjob handler method at the specified schedule with the requests defined in theendowment:cronjob permission.

note For MetaMask to call the Snap'sonCronjob method, you must request theendowment:cronjob permission.

Parameters

An object containing an RPC request specified in theendowment:cronjob permission.

Example

- TypeScript
- JavaScript

index.ts import

type

{ OnCronjobHandler }

from

"@metamask/snaps-sdk" ;

export

const onCronjob :

OnCronjobHandler

=

async

({ request })

=>

{ switch

(request . method)

{ case

"exampleMethodOne" : return snap . request ({ method :

"snap_notify" , params :

{ type :

"inApp" , message :

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

default : throw

new

Error ("Method not found.") ; } } ; index.js module . exports . onCronjob

=

async

({ request })

=>

{ switch

(request . method)

{ case

"exampleMethodOne" : return snap . request ({ method :

"snap_notify" , params :

{ type :

"inApp" , message :

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

default : throw

new

Error ("Method not found.") ; } } ;

onHomePage

To build an embedded UI in MetaMask that any user can access through the Snaps menu, a Snap must expose theonHomePage entry point. MetaMask calls theonHomePage handler method when the user selects the Snap name in the Snaps menu.

note For MetaMask to call the Snap'sonHomePage method, you must request theendowment:page-home permission.

Parameters

None.

Returns

One of the following:

- Acontent
- object displayed usingcustom UI
- .
- Anid
- returned bysnap_createInterface
- forinteractive UI
- .

Example

- TypeScript
- JavaScript

index.ts import

type

{ OnHomePageHandler }

from

"@metamask/snaps-sdk" ; import

{ panel , text , heading }

from

"@metamask/snaps-sdk" ;

export

const onHomePage :

OnHomePageHandler

=

async

( )

=>

{ return

{ content :

panel ([heading ("Hello world!") , text ("Welcome to my Snap home page!") ,]) , } ; } ; index.js import

{ panel , text , heading }

from

"@metamask/snaps-sdk" ;

module . exports . onHomePage

=

async

( )

=>

{ return

{ content :

panel ([heading ("Hello world!") , text ("Welcome to my Snap home page!") ,]) , } ; } ;

onInstall

To implement alifecycle hook that runs an action upon installation, a Snap must expose theonInstall entry point. MetaMask calls theonInstall handler method after the Snap is installed successfully.

note For MetaMask to call the Snap'sonInstall method, you must request theendowment:lifecycle-hooks permission.

Parameters

None.

Example

- TypeScript
- JavaScript

index.ts import

type

{ OnInstallHandler }

from

"@metamask/snaps-sdk" ; import

{ heading , panel , text }

from

"@metamask/snaps-sdk" ;

export

const onInstall :

OnInstallHandler

=

async

( )

=>

{ await snap . request ({ method :

"snap_dialog" , params :

{ type :

"alert" , content :

panel ([heading ("Thank you for installing my Snap") , text ("To use this Snap, visit the companion dapp at metamask.io."
,) ,]) , } , }) ; } ; index.js import

{ heading , panel , text }

from

"@metamask/snaps-sdk" ;

module . exports . onInstall

=

async

( )

=>

{ await snap . request ({ method :

"snap_dialog" , params :

{ type :

"alert" , content :

panel ([heading ("Thank you for installing my Snap") , text ("To use this Snap, visit the companion dapp at metamask.io."
,) ,]) , } , }) ; } ;

onNameLookup

Flask Only This feature is experimental and only available in MetaMask Flask , the canary distribution of MetaMask. To provide custom name resolution , a Snap must export onNameLookup . Whenever a user types in the send field, MetaMask calls this method. MetaMask passes the user input to the onNameLookup handler method.

note For MetaMask to call the Snap's onNameLookup method, you must request the endowment:name-lookup permission.

Parameters

An object containing:

- chainId
-
  - TheCAIP-2
- chain ID.
- address
- ordomain
-
  - One of these parameters is defined, and the other is undefined.

Example

- TypeScript
- JavaScript

```
index.ts import

type

{ OnNameLookupHandler }

from

"@metamask/snaps-sdk" ;

export

const onNameLookup :

OnNameLookupHandler

=

async

( request )

=>

{ const

{ chainId , address , domain }

= request ;

if

( address )

{ const shortAddress = address . substring ( 2 ,

5 ) ; const chainIdDecimal =

parseInt ( chainId . split ( ":" ) [ 1 ] ,

10 ) ; const resolvedDomain =

{ shortAddress } . { chainIdDecimal } .test.domain  ; return

{ resolvedDomains :

[ { resolvedDomain , protocol :

"test protocol"

} ]

} ; }

if

( domain )

{ const resolvedAddress =
"0xc0ffee254729296a45a3885639AC7E10F9d54979" ; return
{ resolvedAddresses :
[ { resolvedAddress , protocol :
"test protocol"
} ] , } ; }
return
null ; } ; index.js module . exports . onNameLookup
=
async
( { request } )
=>
{ const
{ chainId , address , domain }
= request ;
if
( address )
{ const shortAddress = address . substring ( 2 ,
5 ) ; const chainIdDecimal =
parseInt ( chainId . split ( ":" ) [ 1 ] ,
10 ) ; const resolvedDomain =
{ shortAddress } . { chainIdDecimal } .test.domain  ; return
{
resolvedDomains :
[ { resolvedDomain ,
protocol :
"test protocol"
} ]
} ; }
if
( domain )
{ const resolvedAddress =
"0xc0ffee254729296a45a3885639AC7E10F9d54979" ; return
{ resolvedAddresses :
[ { resolvedAddress ,
protocol :
"test protocol"
} ] , } ; }
```

return

null ; } ;

onRpcRequest

To implement acustom JSON-RPC API to communicate with dapps and other Snaps, a Snap must expose theonRpcRequest entry point. Whenever the Snap receives a JSON-RPC request, MetaMask calls theonRpcRequest handler method.

note For MetaMask to call the Snap'sonRpcRequest method, you must request theendowment:rpc permission.

Parameters

An object containing:

- origin
-
    - The origin as a string.
- request
-
    - The JSON-RPC request.

Returns

A promise containing the return of the implemented method.

Example

- TypeScript
- JavaScript

index.ts import

type

{ OnRpcRequestHandler }

from

'@metamask/snaps-sdk' ;

export

const onRpcRequest :

OnRpcRequestHandler

=

async

({ origin , request , })

=>

{ switch

(request . method)

{ case

'hello' : return

'world!' ;

default : throw

new

Error ('Method not found.') ; } } ; index.js module . exports . onRpcRequest

=

async

({ origin , request })

=>

{ switch

(request . method)

{ case

'hello' : return

'world!' ;

default : throw

new

Error ('Method not found.') ; } } ;

onSignature

Flask Only This feature is experimental and only available inMetaMask Flask , the canary distribution of MetaMask. To providesignature insights before a user signs a message, a Snap must expose theonSignature entry point. Whenever asigning method is called, such aspersonal_sign oreth_signTypedData_v4 , MetaMask passes the raw unsigned signature payload to theonSignature handler method.

note For MetaMask to call the Snap'sonSignature method, you must request theendowment:signature-insight permission.

Parameters

An object containing:

- signature
-
  - The raw signature payload.
- signatureOrigin
-
  - The signature origin ifallowSignatureOrigin
- is set totrue
- .

Returns

- An optionalseverity
- property that, if present, must be set toSeverityLevel.Critical
- .
- A content object displayed usingcustom UI
- after the user
- selects theSign
- button.
- Due to current limitations of MetaMask's signature confirmation UI, the content will only be displayed if
- theseverity
- property is present and set toSeverityLevel.Critical
- .

Example

- TypeScript
- JavaScript

index.ts import

```
type
{ OnSignatureHandler , SeverityLevel }
from
"@metamask/snaps-sdk" ; import
{ panel , heading , text }
from
"@metamask/snaps-sdk" ;
export
const onSignature :
OnSignatureHandler
=
async
( { signature , signatureOrigin , } )
=>
{ const insights =
/ Get insights / ; return
{ content :
panel ( [ heading ( "My Signature Insights" ) , text ( "Here are the insights:" ) , ... ( insights . map ( ( insight )
=>
text ( insight . value ) ) ) , ] ) , severity : SeverityLevel . Critical , } ; } ; index.js import
{
SeverityLevel
}
from
"@metamask/snaps-sdk" ; import
{ panel , heading , text }
from
"@metamask/snaps-sdk" ;
module . exports . onSignature
=
async
( { signature , signatureOrigin , } )
=>
{ const insights =
/ Get insights / ; return
{ content :
panel ( [ heading ( "My Signature Insights" ) , text ( "Here are the insights:" ) , ... ( insights . map ( ( insight )
```

=>

text (insight . value))) ,]) , severity :

SeverityLevel . Critical , } ; } ;

onTransaction

To provide transaction insights before a user signs a transaction, a Snap must expose theonTransaction entry point. Whenever there's a contract interaction, and a transaction is submitted using the MetaMask extension, MetaMask calls theonTransaction handler method. MetaMask passes the raw unsigned transaction payload toonTransaction .

note For MetaMask to call the Snap'sonTransaction method, you must request the[endowment:transaction-insight]() permission.

Parameters

An object containing:

- transaction
-
    - The raw transaction payload.
- chainId
-
    - The[CAIP-2]()
- chain ID.
- transactionOrigin
-
    - The transaction origin if[allowTransactionOrigin]()
- is set totrue
- .

Returns

One of the following:

- Acontent
- object displayed using[custom UI]()
- , alongside the confirmation
- for the transaction thatonTransaction
- was called with.
- Anid
- returned by[snap_createInterface]()
- for[interactive UI]()
- .

Example

- TypeScript
- JavaScript

index.ts import

type

{ OnTransactionHandler }

from

"@metamask/snaps-sdk" ; import

{ panel , heading , text }

from

"@metamask/snaps-sdk" ;

export

const onTransaction :

OnTransactionHandler

=

async

({ transaction , chainId , transactionOrigin , })

=>

{ const insights =

/ *Get insights* / ; return

{ content :

panel ([heading ("My Transaction Insights") , text ("Here are the insights:") , ... (insights . map ((insight)

=>

text (insight . value))) ,]) , } ; } ; index.js import

{ panel , heading , text }

from

"@metamask/snaps-sdk" ;

module . exports . onTransaction

=

async

({ transaction , chainId , transactionOrigin , })

=>

{ const insights =

/ *Get insights* / ; return

{ content :

panel ([heading ("My Transaction Insights") , text ("Here are the insights:") , ... (insights . map ((insight)

=>

text (insight . value))) ,]) , } ; } ;

Transaction severity level

Flask Only This feature is experimental and only available inMetaMask Flask , the canary distribution of MetaMask. This feature enables transaction insight Snaps to return an optional severity level ofcritical . MetaMask shows a modal with the warning before the user can confirm the transaction. Using the previous example foronTransaction , the following code adds a single line to return an insight with the severity levelcritical :

- TypeScript
- JavaScript

index.ts import

type

{ OnTransactionHandler }

from

"@metamask/snaps-sdk" ; import

{ panel , heading , text }

from

"@metamask/snaps-sdk" ;

export

const onTransaction :

OnTransactionHandler

=

async

({ transaction , chainId , transactionOrigin , })

=>

{ const insights =

/ *Get insights* / ; return

{ content :

panel ([heading ("My Transaction Insights") , text ("Here are the insights:") , ... (insights . map ((insight)

=>

text (insight . value))) ,]) , severity :

"critical" , } ; } ; index.js import

{ panel , heading , text }

from

"@metamask/snaps-sdk" ;

module . exports . onTransaction

=

async

({ transaction , chainId , transactionOrigin , })

=>

{ const insights =

/ *Get insights* / ; return

{ content :

panel ([heading ("My Transaction Insights") , text ("Here are the insights:") , ... (insights . map ((insight)

=>

text (insight . value))) ,]) , severity :

"critical" , } ; } ;

onUpdate

To implement alifecycle hook that runs an action upon update, a Snap must expose theonUpdate entry point. MetaMask calls theonUpdate handler method after the Snap is updated successfully.

note For MetaMask to call the Snap'sonUpdate method, you must request theendowment:lifecycle-hooks permission.

Parameters

None.

Example

- TypeScript
- JavaScript

index.ts import

type

{ OnUpdateHandler }

from

"@metamask/snaps-sdk" ; import

{ heading , panel , text }

from

"@metamask/snaps-sdk" ;

export

const onUpdate :

OnUpdateHandler

=

async

( )

=>

{ await snap . request ({ method :

"snap_dialog" , params :

{ type :

"alert" , content :

panel ([heading ("Thank you for updating my Snap") , text ("New features added in this version:" ,) , text ("Added a dialog that appears when updating.") ,]) , } , }) ; } ; index.js import

{ heading , panel , text }

from

"@metamask/snaps-sdk" ;

module . exports . onUpdate

=

async

( )

=>

{ await snap . request ({ method :

"snap_dialog" , params :

{ type :

"alert" , content :

panel ([heading ("Thank you for updating my Snap") , text ("New features added in this version:" ,) , text ("Added a dialog that appears when updating.") ,]) , } , }) ; } ;

onUserInput

Flask Only This feature is experimental and only available in[MetaMask Flask](#) , the canary distribution of MetaMask. To respond to[interactive UI](#) events, a Snap must exportonUserInput .

Parameters

- id
-
  - The ID of the interface being acted on.
- event
-
  - An event object containing:* type
-
  -
    - The type of the event.
-
  - Possible values areButtonClickEvent
-
  - ,FormSubmitEvent
-
  - , orInputChangeEvent
-
  - .
-
  - These enums are exported from the@metamask/snaps-sdk
-
  - module.
-
  - name
-
  -
    - The name of the component that fired the event.
-
  - Optional when the event type isButtonClickEvent
-
  - .
-
  - value
-
  -
    - When the event type isFormSubmitEvent
-
  - , the values in the form as an object.

Example

- TypeScript
- JavaScript

index.ts import

type

{ OnUserInputHandler }

from

"@metamask/snaps-sdk" ; import

{ UserInputEventType }

from

"@metamask/snaps-sdk" ;

```
export
const onUserInput :
OnUserInputHandler
=
async
( { id , event } )
=>
{ if
( event . type === UserInputEventType . FormSubmitEvent )
{ console . log ( "The submitted form values are" , event . value ) ; } } ; index.js const
{
UserInputEventType
}
=
require ( "@metamask/snaps-sdk" ) ;
module . exports . onUserInput
=
async
( { id , event } )
=>
{ if
( event . type
===
UserInputEventType . FormSubmitEvent )
{ console . log ( "The submitted form values are" , event . value ) ; } } ;
```

Edit this page