Jest API and options

You can use the@metamask/snaps-jest package forend-to-end Snaps testing . This reference describes the availableAPI methods ,Jest matchers , andoptions .

API methods

installSnap

Installs a Snap in the execution environment. We recommend using this function in each test to ensure that it starts with a clean slate.

Parameters

By default, if the built-in server is enabled,installSnap installs the Snap from the built-in server. Otherwise, you must specify a Snap ID to install.

Returns

An object with functions that can be used to interact with the Snap.

Example

import

{ installSnap }

from

"@metamask/snaps-jest" ;

describe ("MySnap" ,

( )

=>

{ it ("should do something" ,

async

( )

=>

{ await

installSnap (/ *optional Snap ID*/) ; // ... }) ; }) ;

request

Sends a JSON-RPC request to the Snap.

Parameters

A JSON-RPC request object with an addition optionalorigin property.

Returns

A promise that resolves to the response from theonRpcRequest entry point, which can be checked usingJest matchers .

Example

import

{ installSnap }

from

"@metamask/snaps-jest" ;

describe ("MySnap" ,

( )

=>

{ it ("should respond to foo with bar" ,

async

( )

=>

{ const

{ request }

=

await

installSnap (/ *Optional snap ID* /) ; const response =

await

request ({ origin :

"http://localhost:8080" , method :

"foo" , params :

[ ] , }) ;

/ *Check the response using Jest matchers. Since the response is a standard JSON-RPC response, * you can use any standard Jest matchers to check it, including snapshot matchers.* / expect (response) . toRespondWith ("bar") ; expect (response) . not . toRespondWithError ("baz") ; expect (response) . toMatchSnapshot ( ) ; }) ; }) ;

onTransaction

Sends a transaction to the Snap.

Parameters

A transaction object with the following properties:

- origin
- chainId
- from
- to
- value
- data
- gasLimit
- maxFeePerGas
- maxPriorityFeePerGas
- nonce

All properties are optional. The addresses are randomly generated by default. Most values can be specified as a hex string or a decimal number.

Returns

An object with the user interface that was shown by the Snap, in the onTransaction entry point.

Example

```
import
{ installSnap }
from
"@metamask/snaps-jest" ; import
{ panel , text }
from
"@metamask/snaps-sdk" ;
describe ( "MySnap" ,
( )
=>
{ it ( "should return insights" ,
async
( )
=>
{ const
{ onTransaction }
=
await
installSnap ( / Optional Snap ID / ) ; const response =
await
onTransaction ( { value :
"0x0" , data :
"0x" , gasLimit :
"0x5208" , maxFeePerGas :
"0x5208" , maxPriorityFeePerGas :
"0x5208" , nonce :
"0x0" , } ) ;
expect ( response ) . toRender ( panel ( [ text ( "Hello, world!" ) ] ) ) ; } ) ; } ) ;
```

onCronjob

Runs a cronjob in the Snap. The request is normally specified in the Snap manifest file under the endowment:cronjob permission, but this method allows you to run cronjobs that are not specified in the manifest as well.

Parameters

A JSON-RPC request object.

Returns

A promise that resolves to the response from the onCronjob entry point, which can be checked using Jest matchers .

Example

```
import
{ installSnap }
from
"@metamask/snaps-jest" ;
describe ( "MySnap" ,
( )
=>
{ it ( "should end foo cronjobs with response bar" ,
async
( )
=>
{ const
{ onCronjob }
=
await
installSnap ( / Optional snap ID / ) ; const response =
await
onCronjob ( { method :
"foo" , params :
[ ] , } ) ;
// Check the response using Jest matchers. expect ( response ) . toRespondWith ( "bar" ) ; expect ( response ) . not . toRespondWithError ( "baz" ) ; } ) ; } ) ;
```

onHomePage

Requests the home page of the Snap. It takes no arguments, and returns a promise that resolves to the response from the onHomePage entry point.

```
import
{ installSnap }
from
"@metamask/snaps-jest" ; import
{ panel , text }
from
"@metamask/snaps-sdk" ;
describe ( "MySnap" ,
( )
=>
```

{ it ("should render the home page" ,

async

( )

=>

{ const

{ onHomePage }

=

await

installSnap (/ *Optional snap ID* /) ; const response =

await

onHomePage ( ) ;

expect (response) . toRender (panel ([text ("Hello, world!")])) ; }) ; }) ;

getInterface

If your Snap usessnap_dialog to show user interfaces, you can use therequest.getInterface method to interact with the user interfaces. This method is present on the return value of therequest method.

Returns

This method waits for the user interface to be shown, and returns an object with functions that can be used to interact with the user interface.

Example

import

{ installSnap }

from

"@metamask/snaps-jest" ; import

{ text }

from

"@metamask/snaps-sdk" ; import

{ assert }

from

"@metamask/utils" ;

describe ("MySnap" ,

( )

=>

{ it ("should render an alert with hello world" ,

async

( )

=>

{ const

{ request }

=

await

installSnap (/ *Optional Snap ID* /) ;

// Note: You cannot resolve the promise yet! const response =

request ({ method :

"foo" , }) ;

const ui =

await response . getInterface ( ) ;

// This is useful if you're using TypeScript, since it infers the type of the user interface. assert (ui . type

===

"alert") ; expect (ui) . toRender (text ("Hello, world!")) ;

// Select the OK button. await ui . ok ( ) ;

// Now you can resolve the promise. const result =

await response ; expect (result) . toRespondWith ("bar") ; }) ; }) ;

Jest matchers

@metamask/snaps-jest includes the following Jest matchers that you can use to assert that a response from a Snap matches an expected value:

- toRespondWith(expectedResponse)
-
    - Checks if a response matches an expected response.
- This matcher checks theresult
- property of the response.
- If the response is an error, this matcher fails.
- toRespondWithError(expectedError)
-
    - Checks if a response matches an expected error.
- This matcher checks theerror
- property of the response.
- If the response is not an error, this matcher fails.
- toSendNotification(notificationText)
-
    - Checks if a Snap sent a notification.
- toRender(expectedInterface)
-
    - Checks if a Snap rendered an interface.
- This is useful for testing the user interface of a Snap, either for a[snap_dialog](#)
- or a user interface rendered by the[transaction insights API](#)
- .

Options

You can pass the following options when[configuring@metamask/snaps-jest](#) . All options are optional.

server

Options for the built-in HTTP server included with this package. This server serves the execution environment, simulator, and the Snap bundle during tests.

The server options are:

- enabled
-
    - Enables or disables the built-in HTTP server.
- Set tofalse
- to use your own HTTP server, which you can specify when callinginstallSnap
- , e.g.installSnap("local:http://my-server")
- .
- The default istrue
- .
- port
-
    - The port to use for the built-in HTTP server.
- The default is a random available (unprivileged) port.
- root
-
    - The path to the root directory to serve the Snap files from.
- This is useful if you want to serve the Snap files from a different directory than the one Jest is
- running from.
- The default is the current working directory.

Example

jest.config.js module . exports

=

{ preset :

"@metamask/snaps-jest" , testEnvironmentOptions :

{ server :

{ port :

8080 , root :

"/path/to/snap/files" , } , } , } ;

Edit this page