Snaps permissions

Snaps canrequest the following permissions .

RPC API permissions

You must request permission to callSnaps API methods . For example, to request to callsnap_dialog , add the following to the manifest file:

snap.manifest.json "initialPermissions" :

{ "snap_dialog" :

{ } }

Endowments

endowment:cronjob

To runcron jobs for the user, a Snap must request theendowment:cronjob permission. This permission allows the Snap to specify cron jobs that trigger theonCronjob entry point.

This permission takes an object with an array ofjobs , each containing two parameters:

- expression
- - Acron expression
- that defines the schedule of the job.
- request
- - A JSON-RPC request object that will be sent to the Snap'sonCronjob
- entry point when
- the job is executed.

tip You can modify the cron job's execution limit usingSnap-defined timeouts . Specify this permission in the manifest file as follows:

snap.manifest.json "initialPermissions" :

{ "endowment:cronjob" :

{ "jobs" :

[{ "expression" :

" * * *" , "request" :

{ "method" :

"exampleMethodOne" , "params" :

{ "param1" :

"foo" } } } , { "expression" :

"*/2 * * *" , "request" :

{ "method" :

"exampleMethodTwo" , "params" :

{ "param1" :

"bar" } } }] } }

endowment:ethereum-provider

To communicate with a node using MetaMask, a Snap must request theendowment:ethereum-provider permission. This permission exposes theethereum global to the Snap execution environment, allowing Snaps to call someMetaMask JSON-RPC API methods. This global is anEIP-1193 provider.

Specify this permission in the manifest file as follows:

snap.manifest.json "initialPermissions" :

{ "endowment:ethereum-provider" :

{ } } note Theethereum global available to Snaps has fewer capabilities thanwindow.ethereum for dapps. See thelist of methods not available to Snaps.

endowment:page-home

To present a dedicated UI within MetaMask, a Snap must request theendowment:page-home permission. This permission allows the Snap to specify a "home page" by exposing theonHomePage entry point. You can use anycustom UI components to build an embedded home page accessible through the Snaps menu.

Specify this permission in the manifest file as follows:

snap.manifest.json "initialPermissions" :

{ "endowment:page-home" :

{ } }

endowment:keyring

For a dapp to callKeyring API methods on an account management Snap, the Snap must configure a list of allowed dapp URLs using theendowment:keyring permission. If a dapp hosted on a domain not listed in theallowedOrigins attempts to call a Keyring API method, MetaMask rejects the request.

tip You can modify the Keyring API's execution limit usingSnap-defined timeouts . Specify this permission in the manifest file as follows:

snap.manifest.json "initialPermissions" :

{ "endowment:keyring" :

{ "allowedOrigins" :

["https://"] } }

endowment:lifecycle-hooks

To implement alifecycle hook that runs an action when a user installs or updates a Snap, the Snap must request theendowment:lifecycle-hooks permission. This permission allows the Snap to expose theonInstall andonUpdate entry points, which MetaMask calls after a successful installation or update, respectively.

tip You can modify the lifecycle hooks' execution limit usingSnap-defined timeouts . Specify this permission in the manifest file as follows:

snap.manifest.json "initialPermissions" :

{ "endowment:lifecycle-hooks" :

{ } }

endowment:name-lookup

Flask Only This feature is experimental and only available inMetaMask Flask , the canary distribution of MetaMask. To providecustom name resolution , a Snap must request theendowment:name-lookup permission. This permission grants the Snap read-only access to user input or an address by exporting theonNameLookup entry point.

This permission takes an object with two optional properties:

- chains
-
    - An array ofCAIP-2
- chain IDs for which the Snap can resolve names and addresses.
- Pass this array to reduce overhead on your Snap by making sure it only receives requests for
- chains it can resolve.
- matchers
-
    - An object that helps reduce overhead by filtering the domains passed to your Snap.
- This must contain at least one of the following properties:* tlds
-
    -
        - An array of strings for top-level domains that the Snap supports.
-
    - schemes
-
    -
        - An array of strings for schemes that the Snap supports.

tip You can modify the name lookup logic's execution limit usingSnap-defined timeouts . Specify this permission in the manifest file as follows:

snap.manifest.json "initialPermissions" :

{ "endowment:name-lookup" :

{ "chains" :

["eip155:1"] , "matchers" :

{ "tlds" :

["crypto"] , "schemes" :

["farcaster"] } } } , In this example, the Snap'sonNameLookup entry point would be called for domains such assomeuser.crypto or schemes such asfarcaster:someuser , as long as the domain resolution is happening on Ethereum Mainnet.

endowment:network-access

To access the internet, a Snap must request theendowment:network-access permission. This permission exposes the globalfetch API to the Snaps execution environment.

caution XMLHttpRequest isn't available in Snaps, and you should replace it withfetch . If your dependencies useXMLHttpRequest , you canpatch it away . Specify this permission in the manifest file as follows:

snap.manifest.json "initialPermissions" :

{ "endowment:network-access" :

{ } }

Same-origin policy and CORS

fetch() requests in a Snap are bound by the browser'ssame-origin policy . Since Snap code is executed in an iframe with thesandbox property, the browser sends anOrigin header with the valuenull with outgoing requests. For the Snap to be able to read the response, the server must send anAccess-Control-Allow-Origin CORS header with the value* ornull in the response.

endowment:rpc

To handle arbitrary JSON-RPC requests, a Snap must request theendowment:rpc permission. This permission grants a Snap access to JSON-RPC requests sent to the Snap, using theonRpcRequest entry point.

This permission requires an object with asnaps ordapps property (or both), to signal if the Snap can receive JSON-RPC requests from other Snaps, or dapps, respectively. The default for both properties isfalse .

tip You can modify the RPC API's execution limit usingSnap-defined timeouts . Specify this permission in the manifest file as follows:

snap.manifest.json "initialPermissions" :

{ "endowment:rpc" :

{ "dapps" :

true , "snaps" :

false } }

Allowed origins

Alternatively, you can specify the caveatallowedOrigins to restrict all requests to specific domains or Snap IDs. Calls from any other origins are rejected.

Specify this caveat in the manifest file as follows:

snap.manifest.json "initialPermissions" :

{ "endowment:rpc" :

{ "allowedOrigins" :

["https://metamask.io" , "https://consensys.io" , "npm:@metamask/example-snap"] } } note If you specifyallowedOrigins , you should not specifydapps orsnaps .

endowment:transaction-insight

To provide transaction insights, a Snap must request theendowment:transaction-insight permission. This permission grants a Snap read-only access to raw transaction payloads, before they're accepted for signing by the user, by exposing theonTransaction entry point.

This permission requires an object with anallowTransactionOrigin property to signal if the Snap should pass thetransactionOrigin property as part of theonTransaction parameters. This property represents the transaction initiator origin. The default isfalse .

tip You can modify the transaction insight logic's execution limit usingSnap-defined timeouts . Specify this permission in the manifest file as follows:

snap.manifest.json "initialPermissions" :

{ "endowment:transaction-insight" :

{ "allowTransactionOrigin" :

true } }

endowment:signature-insight

Flask Only This feature is experimental and only available inMetaMask Flask , the canary distribution of MetaMask. To providesignature insights , a Snap must request theendowment:signature-insight permission. This permission grants a Snap read-only access to raw signature payloads, before they're accepted for signing by the user, by exposing theonSignature entry point.

This permission requires an object with anallowSignatureOrigin property to signal if the Snap should pass thesignatureOrigin property as part of theonSignature parameters. This property represents the signature initiator origin. The default isfalse .

Specify this permission in the manifest file as follows:

snap.manifest.json "initialPermissions" :

{ "endowment:signature-insight" :

{ "allowSignatureOrigin" :

true } } ,

endowment:webassembly

To use WebAssembly, a Snap must request theendowment:webassembly permission. This permission exposes the globalWebAssembly API to the Snap execution environment.

Specify this permission in the manifest file as follows:

snap.manifest.json "initialPermissions" :

{ "endowment:webassembly" :

{ } }

Snap-defined timeouts

Many endowments entail having MetaMask run arbitrary code defined in the Snap. The default execution timeout is 60000 milliseconds, or one minute.

You can modify this execution timeout by adding a caveatmaxRequestTime to the permission. It can take values from5000 (5 seconds) to180000 (3 minutes). For example:

snap.manifest.json "initialPermissions" :

{ "endowment:transaction-insight" :

{ "maxRequestTime" :

10000 } } The following endowments accept this caveat:

- endowment:cronjob
- endowment:keyring
- endowment:lifecycle-hooks
- endowment:name-lookup
- endowment:page-home
- endowment:rpc
- endowment:transaction-insight

Dynamic permissions

eth_accounts

A Snap can request permission to call theeth_accounts MetaMask JSON-RPC API method by callingeth_requestAccounts . Callingeth_requestAccounts requires theendowment:ethereum-provider permission:

- Manifest file
- JavaScript

snap.manifest.json "initialPermissions" :

{ "endowment:ethereum-provider" :

{ } } index.js await ethereum . request ({

"method" :

"eth_requestAccounts"

}) ; You can check the presence of the permission by callingwallet_getPermissions . If the permission is present, the result contains a permission with aparentCapability ofeth_accounts . The permission contains arestrictReturnedAccounts caveat,

an array of all the accounts the user allows for this Snap. The following is an exampleeth_accounts permission:

{ "id" :

"47vm2UUi1pccNAeYKGmwF" , "parentCapability" :

"eth_accounts" , "invoker" :

"npm:SNAP_ID" , "caveats" :

[{ "type" :

"restrictReturnedAccounts" , "value" :

["0xc403b37bf1e700cb214ea1be9de066824b420de6"] }] , "date" :

1692616452846 } The user can revoke this permission by going to the Snap's settings underSnap permissions .

Edit this page