Data storage

You can store and manage sensitive information within a Snap using encrypted storage, or non-sensitive information using unencrypted storage. Use thesnap_manageState API method to persist up to 100 MB of data to the user's disk and retrieve it at will. We recommend using this method for storing data in a Snap long term.

Steps

1. Get permission to store data

Request thesnap_manageState permission. Add the following to your Snap's manifest file:

snap.manifest.json "initialPermissions" :

{ "snap_manageState" :

{ } }

2. Use encrypted storage

By default,snap_manageState automatically encrypts data using a Snap-specific key before storing it on the user's disk, and automatically decrypts it when retrieved. This is useful to store sensitive information, such as passwords.

The following example usessnap_manageState to store some data using theupdate operation, and retrieves the data at a later time using theget operation. When the data is no longer required, the Snap's state is cleared using theclear operation.

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

"snap_manageState" , params :

{ operation :

"update" , newState :

{

hello :

"world"

} , } , }) ;

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

await snap . request ({ method :

"snap_manageState" , params :

{

operation :

"get"

} , }) ;

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

// If data storage is no longer necessary, clear it. await snap . request ({ method :

"snap_manageState" , params :

{ operation :

"clear" , } , }) ; tip Accessing encrypted state requires MetaMask to be unlocked. If you need to access encrypted state in a background task such as acron job , usesnap_getClientStatus to ensure that MetaMask is unlocked before accessing state, preventing an unexpected password request.

3. Use unencrypted storage

To use unencrypted storage, setencrypted tofalse when storing, retrieving, or clearing data usingsnap_manageState . The Snap will use a storage section separate from the encrypted storage, and will not encrypt the data. This is useful to access non-sensitive data from background operations such ascron jobs , without requiring the user to enter their password in the case that MetaMask is locked.

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

"snap_manageState" , params :

{ operation :

"update" , newState :

{

hello :

"world"

} , encrypted :

false , } , }) ;

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

await snap . request ({ method :

"snap_manageState" , params :

{ operation :

"get" , encrypted :

false , } , }) ;

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

// If data storage is no longer necessary, clear it. await snap . request ({ method :

"snap_manageState" , params :

{ operation :

"clear" , encrypted :

false , } , }) ;

Example

See the@metamask/manage-state-example-snap package for a full example of storing data usingsnap_manageState . This example exposes acustom JSON-RPC API for dapps to store, retrieve, and clear data.

Edit this page