Allow automatic connections

You can authorize specific dapps or Snaps to automatically connect to your Snap, skipping the need for users to manually confirm a connection when the dapp or Snap callswallet_requestSnaps . This is useful, for example, forSnaps used by multiple dapps orSnaps installed from the Snaps directory .

Steps

1. Add authorized dapps ininitialConnections

Add aninitialConnections field in the Snap manifest file, specifying each dapp you want to allow to automatically connect. For example:

snap.manifest.json "initialConnections" :

{ "https://voyager-snap.linea.build" :

{ } } important initialConnections is not a replacement forendowment:rpc .endowment:rpc is still required to allow dapps or Snaps to call RPC methods of your Snap. When testing, you can specify the local site. For example:

snap.manifest.json "initialConnections" :

{ "http://localhost:8000" :

{ } } We recommend removing local sites before deploying your Snap to production.

2. Connect to the Snap from an authorized dapp

When you visit a dapp specified in the Snap'sinitialConnections , and the dapp callswallet_requestSnaps , if the Snap is already installed, the dapp connects immediately and can make further calls to the Snap. If the Snap is not installed, you see a confirmation to install the Snap.

Additionally, since the connection between the Snap and a dapp listed ininitialConnections is automatic, callingwallet_getSnaps from the dapp returns the Snap even if the dapp never calledwallet_requestSnaps . This makes the flow of using the Snap from an authorized dapp completely seamless.

Example

The following sample code, adapted from theLinea Voyager Snap companion dapp, shows the flow of connecting to a Snap from a dapp authorized ininitialConnections .

While the code is exactly the same as it would be without the dapp being listed ininitialConnections , the comments show how the flow is different from the user's perspective.

The key is that callingwallet_getSnaps from an authorized dapp when the Snap is already installed returns that Snap even if the dapp never explicitly connected to it by callingwallet_requestSnaps .

script.js const snapId =

"npm:@myorg/mysnap" ;

// This function is called when the EIP-6963 process of finding MetaMask is successful. const

MetaMaskFound

=

async

(providerDetail)

=>

{ const

{ provider }

= providerDetail ;

// This call returns the Snap ID if it is already installed. const snaps =

await provider . request ({ method :

"wallet_getSnaps" , }) ;

if

(Object . keys (snaps) . includes (snapId))

{ // If we're here, the Snap is installed. We're good to go. return ; }

// If we're here, the Snap is not installed. Let's install it. // Since the Snap is not installed, the user still sees a confirmation to install the Snap. try

{ const result =

await provider . request ({ method :

"wallet_requestSnaps" , params :

{ [snapId] :

{ } , } , }) ;

// If we're here, the Snap is installed, and the dapp is successfully connected to it. }

catch

(err)

{ // If we're here, something went wrong. For example, the user might have declined the Snap // installation confirmation. } } ;

When to use initial connections

The following scenarios showcaseinitialConnection 's usefulness, in order from most to least useful:

Snap used by multiple dapps

This flow is as follows:

1. You build a Snap that can be used by multiple dapps, for example, a protocol Snap.
2. You authorize an automatic connection to your Snap's companion dapp by listing it ininitialConnections
3. .
4. A user installs your Snap while interacting with a dapp that uses it.
5. The user lands on your Snap's companion dapp.
6. The user can interact with the companion dapp without having to manually connect to the Snap.

For even greater benefit, consider authorizing an automatic connection for some known dapps that interact with your Snap.

Snap installed from the Snaps directory

This flow is as follows:

1. You build a Snap that is only really useful for one dapp, its companion dapp.
2. You authorize an automatic connection to the companion dapp by listing it ininitialConnections
3. .
4. A user finds your Snap through theSnaps directory
5. and installs it.
6. The user lands on your Snap's companion dapp.
7. The user can interact with the companion dapp without having to manually connect to the Snap.

Snap installed from your companion dapp

This flow is as follows:

1. You build a Snap, and authorize an automatic connection to its companion dapp by listing the dapp
2. ininitialConnections

3. .
4. A user discovers your Snap through its companion dapp.
5. The user must install the Snap, going through the normal flow.

In this case, adding the companion dapp toinitialConnections is not useful.

Edit this page