Signature insights

Flask Only This feature is experimental and only available inMetaMask Flask , the canary distribution of MetaMask. You can provide signature insights before a user signs a message. For example, you can warn the user about potentially dangerous signature requests.

Steps

1. Request permission to display signature insights

Request theendowment:signature-insight permission by adding the following to your Snap's manifest file:

snap.manifest.json { "initialPermissions" :

{ "endowment:signature-insight" :

{ } } } If you need to receive the origin of the signature request, addallowSignatureOrigin to the permission object, and set it totrue :

snap.manifest.json { "initialPermissions" :

{ "endowment:signature-insight" :

{ "allowSignatureOrigin" :

true } } } When requesting this permission, the following displays in the MetaMask UI when a user installs the Snap:

2. ImplementonSignature

and export it fromindex.ts

Expose anonSignature entry point, which receives asignature object. The shape of this object depends on the chain and the signing method used. This is why it's typed asRecord .

For Ethereum and Ethereum-compatible chains, thesignature object can have one of the following shapes, depending on the signing method used:

- eth_sign
- personal_sign
- eth_signTypedData
- eth_signTypedData_v3
- eth_signTypedData_v4

interface

EthSignature

{ from :

string ; data :

string ; signatureMethod :

"eth_sign" ; } interface

PersonalSignature

{ from :

string ; data :

string ; signatureMethod :

"personal_sign" ; } interface

SignTypedDataSignature

{ from :

string ; data : Record < string ,

any

[ ] ; signatureMethod :

"eth_signTypedData" ; } interface

SignTypedDataV3Signature

{ from :

string ; data : Record < string ,

any

; signatureMethod :

"eth_signTypedData_v3" ; } interface

SignTypedDataV4Signature

{ from :

string ; data : Record < string ,

any

; signatureMethod :

"eth_signTypedData_v4" ; } Your Snap should usesignatureMethod as the source of truth to identify the signature scheme it is providing insights for.

Once you've identified the signature object, your Snap can run any logic, including calling APIs. Then, your Snap must either returnnull if it has no insights to provide, or an object with acontent property and an optionalseverity property as specified in theonSignature entry point.

caution Due to current MetaMask UI limitations, signature insights will only be displayed if your Snap's logic deems the signature to be one that a user shouldn't sign, that is, if you return a severity level ofSeverityLevel.Critical . The following is an example implementation ofonSignature :

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 based on custom logic* / ; return

{ content :

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

=>

text (insight . value))) ,]) , severity : SeverityLevel . Critical , } ; } ; When your Snap returns a signature insight with aseverity ofSeverityLevel.Critical , the custom UI displays in a modal after the user selects theSign button. For example:

Example

See the@metamask/signature-insights-example-snap package for a full example of implementing signature insights.

Edit this page