CoW Hook dApp

What are CoW Hooks?

Technical specification

Hook dApp - is an application that allows you to conveniently program an order to perform any action before or after a swap.

In the CoW Swap interface you can find several hooks for different tasks. For example, you can claim GNO from Gnosis validators before your swap. You can also specify an arbitrary call to a smart contract before or after your swap using the "Build your own hooks" dApp.

But that's not all. You can also develop your own application that will install a hook in CoW Swap! For this purpose, there is a custom application based oniframe .

In order to add a custom hook, you need to:

- Click "add Pre/Post-Hook Action"
- Go to "My Custom Hooks" tab
- Paste a URL of the hook dApp

CoW Hooks are still under development! The feature is not enabled in tbe production environment just yet. You can test CoW Hooks athttps://barn.cow.fi .

How to develop CoW Hook dApps

CoW Hook dApp is a web application that communicates with CoW Swap usingpost-messages . For your convenience there is a npm library@cowprotocol/hook-dapp-lib which provides everything necessary to get started with your hook dApp development.

Install

yarn add @cowprotocol/hook-dapp-lib npm install @cowprotocol/hook-dapp-lib It provides:

- EIP-1193
- provider to interact with a user wallet
- HookDappContext
- which contains environment parameters (chainId, account, etc.) and current order parameters (sell token, validTo, etc.)

And it expects calling following callbacks:

- addHook
- when a hook data is ready and can be added to an order
- editHook
- when hook parameters were changed (edit mode)
- setSellToken / setBuyToken
- if you want to change the trade assets

Quick start

Let's create a simple hook dApp that checks the COW token balance of the order creator.

< html

< body

< button

id

" actionButton "

Add or edit hook </ button

```
< script
        import
{ initCoWHookDapp ,
HookDappContext
}
from
'@cowprotocol/hook-dapp-lib'
// Encoded call of ERC20.balanceOf(account) const
CALL_DATA
=
account
=>
0x70a08231000000000000000000000000 { account . slice ( 2 ) }
let
context :
HookDappContext
|
null
=
null
const
{ actions , provider }
=
initCoWHookDapp ( {
onContext :
( _context )
=> context = _context } )
provider . request ( {
method :
'eth_chainId'
} ) . then ( chainId
=>
{ console . log (User is connected to a network with id= { chainId } ) } )
document . getElementById ( 'actionButton' ) . addEventListener ( 'click' ,
( )
=>
{ if
```

```
( ! context )

{ console . log ( 'App is not loaded yet, please wait a bit.' ) return }

// Edit a hook if

( context . hookToEdit )

{ actions . editHook ( { ... context . hookToEdit , hook :

{ ... context . hookToEdit . hook , gasLimit :

'40000' } } ) }

else

{ // Create a hook actions . addHook ( { hook :

{ target :

'0xdef1ca1fb7fbcdc777520aa7f396b4e015f497ab' ,

// COW token callData :

CALL_DATA ( context . account ) , gasLimit :

'32000' } } ) } } ) </ script

        </ body

        </ html
```

Types specification

```
import

{ initCoWHookDapp , CoWHookDappActions , HookDappContext }

from

'@cowprotocol/hook-dapp-lib'

// context: HookDappContext // actions: CoWHookDappActions // provider: EIP-1193 const

{ actions , provider }

=

initCoWHookDapp ( {

onContext ( context : HookDappContext )

{ }

} )
```

HookDappContext

Parameter Type Optional Description chainId number false Current chainId in CoW Swap. account string true The user's account address. If not set, the wallet is not connected yet. orderParams HookDappOrderParams | null false The parameters for the order. If value isnull , the order is not ready yet. SeeHookDappOrderParams for more details. hookToEdit CowHookDetails true CoW Swap supports editing hooks that have been created already. If the parameter is set, then it's in edit mode. SeeCowHookDetails for more details. isSmartContract boolean | undefined true Whether the account is a smart contract.undefined if unknown. isPreHook boolean false Indicates the position of the hook.

CoWHookDappActions

Parameter Type Description addHook (payload: CowHookCreation) => void The callback should be executed once the hook

is ready to be added. editHook (payload: CowHookDetails) => void The callback should be executed when the hook data is changed. setSellToken / setBuyToken (token: { address: string }) => void Optionally you can change the asset for sell/buy.

Manifest

As stated at the beginning of this document, in the CoW Swap interface you can add your application as a custom application. To do this, the application must containmanifest.json file with following structure:

{ "cow_hook_dapp" :

{ "id" :

"06a2747d08f0026f47aebb91ac13172a318eb3f6116f742751e2d83cc61b8753" , "name" :

"Your Hook Dapp" , "descriptionShort" :

"Your Hook Dapp short description" , "description" :

"Your Hook Dapp full description" , "version" :

"0.0.1" , "website" :

"https://your-cow-hook.dapp" , "image" :

"http://your-cow-hook.dapp/logo.png" , "conditions" :

{ "position" :

"pre" , "smartContractWalletSupported" :

false , "supportedNetworks" :

[1 ,

100 ,

42161] } } } Parameter Type Description id string 64-bit hex application identifier(keccak256(YOUR_APP_NAME)) . This value is used to match hooks in an order with the dApp that generated the hook. name string The name of the Hook Dapp. descriptionShort string A short description. description string A full, detailed description of the Hook Dapp. version string The version number of the Hook Dapp, typically following semantic versioning. website string (URL) The URL link to the Hook Dapp's official website. image string (URL) The URL link to the Hook Dapp's logo or representative image. conditions.position pre | post Specifies the execution position of the hook,pre orpost . If not set, then the Hook Dapp supports both positions. conditions.walletCompatibility HookDappWalletCompatibility[] Indicates whether the Hook Dapp supports smart contract wallets or EOAs. conditions.supportedNetworks array ofintegers List of supported network IDs (e.g.,1 for Ethereum mainnet,100 for Gnosis chain,42161 for Arbitrum). If not set, then the Hook Dapp will be available for any network supported by CoW Swap.

Advanced Hook Dapps

For more complex scenarios, such as transferring the buy amount to another smart-contract or other smart-contract related actions, you will probably need CoW Shed:

- CoW Shed
- Permit, Swap & Bridge CoW Hook example

Edit this page Previous CoW Swap Widget Next Building on CoW Protocol