Onboard (older version)

This page is for Onboard v1, which is an older version. For web3-onboard which is the newest, modernized version, see the web3-onboard page. Use the latest version of Onboard at https://github.com/blocknative/web3-onboard . This version is no longer maintained. See our Migration guide here to migrate to web3-onboard from the older version ?

?

Quickstart

It should take less than 5 minutes to get going with Blocknative Onboard.

Create a Blocknative Account

Go to the Account Dashboard at https://explorer.blocknative.com/account and setup an account with an email address. You will receive an email to confirm your account.

Create your API Key

On the Account Dashboard at https://explorer.blocknative.com/account , create an API key with your choice of name or use/rename the Default Key. Consider using different API keys for development, staging, and production releases.

?

Install the widget using npm

```
Copy npm install bnc-onboard
```

Initialize the Library

Use the minimum required config. There are more options available as detailed in the initialization section.

```
Copy importOnboardfrom'bnc-onboard' importWeb3from'web3'

letweb3;

constonboard=Onboard({ dappId:apiKey,// [String] The API key created by step one above networkId:networkId,// [Integer] The Ethereum network ID your Dapp uses. subscriptions:{ wallet:wallet=>{ web3=newWeb3(wallet.provider) } } });
```

The wallet subscription function will be called when the user selects a wallet so you can instantiate web3 with the selected provider.

Onboard User

Ask the user to select a wallet and then check wallet is ready to transact.

```
Copy awaitonboard.walletSelect(); awaitonboard.walletCheck();
```

And you are live!

Screencasts

See the above Quickstart guide in a screencast:

Onboard.js Quickstart Screencast For a deeper dive with additional wallets that builds on the Quick Start, see this screencast:

Additional Wallets Screencast

Example Code

For an example of a basic React demo integration of both Onboard.js and Notify.js , checkout the GitHub repo:https://github.com/blocknative/onboard-notify-react

and the live demo:

https://reactdemo.blocknative.com/

Initialization

Options

```
```

Copy constoptions={ dappId:String, networkId:Number, networkName:String, darkMode:Boolean, blockPollingInterval:Number, subscriptions:{ address:Function, network:Function, balance:Function, wallet:Function }, walletSelect:{ heading:String, description:String, explanation:String, wallets:Array, agreement:{ version:String, termsUrl:String, privacyUrl:String } }, walletCheck:Array }

```

dappId - [optional]

Your uniqueapiKey that identifies your application. You can generate adappId by visiting theBlocknative account page and create a free account. Thedappid enables Onboard to use Blocknative notification services to efficiently maintain address balance state. This isnot required to use Onboard to connect wallets to Dapps. It is required to efficiently get an up to date wallet balance value via thebalance subscription orgetState() calls.

Efficiently getting an up to date wallet balance value is subject to ourAPI Rate Limits .

networkId - [required]

TheEthereum network id that your application runs on. The following values are valid:

- 1
- main
- Ethereum Network mainnet
- 3
- ropsten
- Ethereum Network testnet
- 4
- rinkeby
- Ethereum Network testnet
- 5
- goerli
- Ethereum Network testnet
- 28
- boba-rinkeby
- Boba Network Rinkeby testnet
- 42
- kovan
- Ethereum Network testnet
- 56
- bsc
- Binance Smart Chain Network mainnet
- 100
- xdai
- xDai POA Network mainnet
- 137
- polygon
- Polygon-Matic Network mainnet
- 250
- fantom-opera
- Fantom Network mainnet
- 288
- boba
- Boba Network mainnet
- 4000
- fantom-testnet
- Fantom Network testnet
-

networkName - [optional]

This option allows one to specify a custom network name for all non-supported networks (seenetworkId above for supported networks). For instance, if you setnetworkId: 56 for Binance smart chain and the user is connected to a different network, the wallet select modal will showlocal as the network to switch to. But if you specifynetworkName: "Binance Mainnet" then this will display instead oflocal .

darkMode - [optional] - Default: false

Opt for the Dark color scheme or not.

blockPollingInterval - [optional] - Default:4000

Change the rate at which blocks are polled by provider, default is set to 4 seconds.

hideBranding - [optional]

If nodappId is provided, then "Powered by Blocknative" branding will be displayed on all modals. You can choose to hide the branding by settinghideBranding: true . If you have included adappId then the branding will be hidden by default. If you would like the branding to display when including adappId then sethideBranding: false .

subscriptions - [optional]

An object of callback functions that get called whenever the corresponding value changes. Use these callbacks to sync your UI or to store the values for use elsewhere in your code. Valid parameters to register callbacks for are:
```

- address
-
  - called with the user's current address[string]
- ens
-
  - called with the user's current ens name (if available)[object]
-
  - name
-
  -
    - The user's resolved ens name[string]
-
  - avatar
-
  -
    - A link to the user's avatar[string]
-
  - getText
-
  -
    - A function to retrieve the user's text records[function]
-
  - contentHash
-
  -
    - The user's content hash record[string]
- *
- network
-
  - called with the user's current network id[number]
- balance
-
  - called with the user's current balance[string]
- wallet
-
  - called with the user's current wallet[object]
-
  - provider
-
  -
    - The JavaScript provider for interacting with the wallet
-
  - name
-
  -
    - The wallet display name
-
  - instance
-
  -
    - If the wallet type is'sdk'
-
  - then this is the initialized wallet instance
-
  - type
-
  -
    - The wallet type'hardware' | 'injected' | 'sdk'
-
  - connect
-
  -
    - The function that initiates the wallet connection logic
-
  - dashboard
-
  -
    - Some SDK wallets allow for opening to wallet dashboard
-
  - icons
-
  - -[object]
-
  - Image strings for the wallet icon{ svg, src, srcset }
- *
-

walletSelect - [optional]

A object that defines how the wallet select screen will render using the following parameters:

- heading
-
  - [optional] : A string that will be the wallet select modal heading
- description
-
  - [optional] : A string that will be the wallet select modal description
- explanation
-
  - [optional] : A string that will be included in the "What is a Wallet" section
- wallets
-
  - [optional] : An array ofwallet modules
- orwallet initialization objects
- that the user can select from
- agreement
-
  - [optional]: An object containing properties for use with terms of service agreement
-
  - version
-
  -
    - The current version of the terms that the user needs to agree to. When this value is updated the user's agreement is invalidated and they must re-agree upon next usage.
-
  - termsUrl
-
  -
    - [optional]: A string that will be used for the terms of service url that a user can agree to before using onboard
-
  - privacyUrl
-
  -
    - [optional]: A string that will be used for the privacy policy url that a user can agree to before using onboard
- *
-

If not provided, then the defaultheading anddescription andwallets that don't require initialization will be included for you.

walletCheck - [optional]

An array ofwallet check modules orwallet check initialization objects that will be used to check if the user is ready to transact.

Initialize Onboard

To initialize Onboard, simply import the library and call it with yourinitializationOptions :

```

Copy importOnboardfrom'bnc-onboard'

constinitializationOptions={ // ..... options here }

constonboard=Onboard(initializationOptions)

```

The return value will be the initializedonboard library which contains the API functions below.

Customizing 'unknown' Network Name

If you initialize Onboard with anetworkId that is not one of thesupported networks all modals will refer to that network with the nameunknown . You can name unrecognized networks by also passing in anetworkName in to your initialization object:

```

Copy constonboard=Onboard({ // ... other init options networkId:137, networkName:'matic' })

```

Local Networks

If you are using a local network for testing, you can ensure that the onboarding checks work correctly by setting thenetworkId for that local network to a number that is not taken by other valid networks. If you are using theganache-cli for instance, you can set thenetworkId via the-i flag:

```

Copy ganache-cli-i35

```

This will set the GanachenetworkId to35

Then in your Onboard config, also set thenetworkId property to the same value:

```
Copy constonboard=Onboard({ networkId:35, // ...other config options })
```

API

walletSelect

When you are ready to get a user to select a wallet and connect it to your Dapp, you can call thewalletSelect function to display the wallet select modal:

```
Copy constwalletSelected=awaitonboard.walletSelect() // returns a Promise that: // resolves with true if the user selected a wallet // resolves with false if the user exited from the wallet select modal
```

This function will show a modal that displays buttons for all of the wallets that you initialized onboard with. It will guide the user through the process of connecting to the wallet that they select. Once the process is successful the function will resolve withtrue . This means that theprovider subscription will have been called with the provider of the selected wallet and you can go ahead and instantiate your web3 library with the provider and also instantiate your contracts.

ThewalletSelect function can also be called with aautoSelectWallet string. If the string passed in is a validwalletName , then Onboard will automatically select that wallet for the user. This enables saving the previously selected wallet inlocalStorage and automatically selecting it for the user when they next visit your Dapp.

walletCheck

Once a wallet is selected, you will want to make sure that the user's wallet is prepared and ready to transact by calling thewalletCheck function:

```
Copy constreadyToTransact=awaitonboard.walletCheck() // returns a Promise that: // resolves with true if user is ready to transact // resolves with false if user exited before completing all wallet checks
```

This function will run through thewalletCheck modules that were passed in via initialization sequentially, making sure the user has passed the condition contained in each module and eventually resolves withtrue if the user completed the sequence. This means that the user is ready to transact. This function is useful to call before every transaction to make sure that nothing has changed since the last time it was called.

walletReset

You may want to reset all of Onboard's internal wallet state and also disconnect from any active SDK instances when a user logs out of your app. You can call thewalletReset function to do this easily.

```
Copy // user wants to log out of session and the wallet state needs to be reset... onboard.walletReset() // this method is synchronous and returns undefined
```

accountSelect

If you want to show the account select modal for hardware wallets at any time you can call theaccountSelect function. For example you could display a button that allows the user switch accounts:

```
Copy Switch Account
```

getState

This function will give you the current state of the user:

```
Copy constcurrentState=onboard.getState() console.log(currentState) // { // address: string // network: number // balance: string // wallet: Wallet // mobileDevice: boolean // appNetworkId: number // }
```

config

You can update some configuration parameters by passing aconfig object in to theconfig function:

```
```

Copy onboard.config({ darkMode:true,networkId:4})

```

Available parameters that you can update are:

- darkMode
- -[boolean]
- networkId
- -[number]
- 

Built-in Modules

Wallet Modules

Initialization:

To initialize the built in wallet modules, an array of wallet initialization objects needs to be passed to thewalletSelect.wallets parameter of the Onboard initialization object. The order of the array is the order that the wallets will be displayed. Mobile wallets will be displayed when a user is on a mobile device and desktop wallets are displayed when they are on a desktop device.

Each wallet initialization object has additional optional customization parameters:

- preferred
- -Boolean
- Will define whether it is shown at the top of the selection screen. If any of the wallets are set topreferred
- , then all other wallets that aren't, will be hidden until the user clicks the "Show More" button. If there are no wallets set topreferred
- , then the first four wallets will be shown and the remaining wallets will be hidden until the show more button is clicked.
- label
- -String
- Will over ride the wallet name that is displayed on the button.
- svg
- -String
- Ansvg
- string that will over ride the icon that is displayed on the button.
- iconSrc
- -String
- An alternative to providing thesvg
- string, by providing a url source.
- display
- -Object
- Customize which wallets to display based on which device a user is using. Note: For wallets that are mobile or desktop only this option willnot
- override this. For instance, if a wallet is mobile only settingdisplay: { desktop: true }
- will not display this wallet on desktop.
- 
    - desktop
- 
    - -Boolean
- 
    - default:true
- 
    - mobile
- 
    - -Boolean
- 
    - default:true
- *
- 

An example configuration that includes all of the supported wallets:

```

Copy importOnboardfrom'bnc-onboard'

constFORTMATIC_KEY="Your Fortmatic key here" constPORTIS_KEY="Your Portis key here" constINFURA_KEY="Your Infura key here" constAPP_URL="Your app url here" constCONTACT_EMAIL="Your contact email here" constRPC_URL="https://.infura.io/v3/" constAPP_NAME="Your app name here"

constwallets=[{ walletName:"coinbase",preferred:true}, { walletName:"trust",preferred:true,rpcUrl:RPC_URL}, { walletName:"metamask",preferred:true}, { walletName:"authereum"}, { walletName:'trezor', appUrl:APP_URL, email:CONTACT_EMAIL, rpcUrl:RPC_URL }, { walletName:'ledger', rpcUrl:RPC_URL }, { walletName:'lattice', rpcUrl:RPC_URL, appName:APP_NAME }, { walletName:'keepkey', rpcUrl:RPC_URL }, { walletName:'mewwallet', rpcUrl:RPC_URL }, { walletName:'cobovault', rpcUrl:RPC_URL, appName:APP_NAME, }, { walletName:'keystone', rpcUrl:RPC_URL, appName:APP_NAME, }, { walletName:"fortmatic", apiKey:FORTMATIC_KEY, preferred:true }, { walletName:"portis", apiKey:PORTIS_KEY, preferred:true, label:'Login with Email' }, { walletName:"walletConnect", infuraKey:INFURA_KEY }, { walletName:"opera"}, { walletName:"operaTouch"}, { walletName:"torus"}, { walletName:"status"}, { walletName:"walletLink"rpcUrl:RPC_URL,appName:APP_NAME} { walletName:"imToken",rpcUrl:RPC_URL}, { walletName:"meetone"}, { walletName:"mykey",rpcUrl:RPC_URL}, { walletName:"huobiwallet",rpcUrl:RPC_URL}, { walletName:"hyperpay"}, { walletName:"wallet.io",rpcUrl:RPC_URL}, { walletName:"atoken"}, { walletName:"frame"}, { walletName:"ownbit"}, { walletName:"alphawallet"}, { walletName:"gnosis"}, { walletName:"xdefi"}, { walletName:"bitpie"}, { walletName:"binance"}, { walletName:"liquality"}, { walletName:"tally"}, {

walletName:"blankwallet"}, { walletName:"mathwallet"}, { walletName:"1inch"}, { walletName:"ronin"},]

constonboard=Onboard({ //... other options walletSelect:{ wallets:wallets } })

```

The following list of wallet modules are included in Onboard and theirinitializationObject is listed for each wallet:

Wallets

- detectedwallet
- 

The detected wallet will be displayed as an option when an injected Ethereum provider is detected, but it is not recognized by Onboard. This allows users to connect to your app with any provider that conforms to the EIP-1193 standard.

- metamask - (desktop & mobile support)
- 

```

Copy { walletName:'metamask'}

```

- authereum - (desktop & mobile support)
- 

```

Copy { walletName:'authereum', disableNotifications:false// [OPTIONAL] rpcUri:''// [OPTIONAL] webUri:''// [OPTIONAL] xsUri:''// [OPTIONAL] blockedPopupRedirect:true// [OPTIONAL] }

```

- walletConnect - (desktop & mobile support)
- 

```

Copy { walletName:'walletConnect', infuraKey:'INFURA_KEY',// your infura id [String][Optional if rpc is provided instead] rpc:{

},// [Optional] bridge:''// url to a bridge server [String][Optional] }

```

- portis - (desktop & mobile support)
- 

```

Copy { walletName:'portis', apiKey:'PORTIS_KEY'// your Portis apiKey [String] }

```

- fortmatic - (desktop & mobile support)
- 

```

Copy { walletName:'fortmatic', apiKey:'FORTMATIC_KEY'// your Fortmatic apiKey [String] rpcUrl:''// optional - use only for custom networks such as matic }

```

- trust - (mobile only support)
- 

```

Copy { walletName:'trust'}

```

- coinbase - (mobile only support)
- 
  - Desktop support for``
- 
  - Coinbase Wallet
- 
  - can be found through
- 
  - walletLink module
- *
-
```

```

Copy { walletName:'coinbase'}

```

- opera - (desktop & mobile support)
-

```

Copy { walletName:'opera'}

```

- opera touch - (mobile only support)
-

```

Copy { walletName:'operaTouch'}

```

- status - (mobile only support)
-

```

Copy { walletName:'status'}

```

- torus - (desktop & mobile support)
-

For additional details on the Torus options, check out their docs

```

Copy { walletName:'torus', rpcUrl:String,// [optional] buttonPosition:String,// [optional] modalZIndex:Number,// [optional] apiKey:String,// [optional] buildEnv:String,// 'production' | 'development' | 'staging' | 'testing' | 'lrc' [optional] enableLogging:Boolean,// [optional] enabledVerifiers:VerifierStatus,// [optional] loginConfig:LoginConfig,// [optional] showTorusButton:Boolean,// [optional] integrity:IntegrityParams,// [optional] whiteLabel:WhiteLabelParams,// [optional] loginMethod:String// 'google' | 'facebook' | 'twitch' | 'reddit' | 'discord' | [optional] }

```

- trezor - (desktop & mobile support)
-

Trezor requires extra details such asappUrl andemail for their manifest

```

Copy { walletName:'trezor', appUrl:APP_URL,// the url of your app (required for manifest) email:CONTACT_EMAIL,// your contact email, (required for manifest) rpcUrl:RPC_URL,// url to connect to an RPC endpoint (ie infura) // See section Hardware Wallet Custom Networks for more info customNetwork:HardwareWalletCustomNetwork }

```

- ledger - (desktop & mobile support)
-

Ledger has an additional optionalLedgerTransport parameter for when you are building an Electron app and need to pass in the node ledger transport module . You can also use this option to pass in whichever Ledger transport that you would like to use based on feature detection (eg. Windows 10 user might need WebUSB transport instead of the default U2F transport).

```

Copy { walletName:'ledger', rpcUrl:RPC_URL// url to connect to an RPC endpoint (ie infura) LedgerTransport:TransportNodeHid, // See section Hardware Wallet Custom Networks for more info customNetwork:HardwareWalletCustomNetwork }

```

- lattice - (desktop & mobile support)
-

```

Copy { walletName:'lattice', rpcUrl:RPC_URL,// url to connect to an RPC endpoint (ie infura) appName:APP_NAME// app name that is displayed on device }

```

- keepkey - (desktop only support)
-

```
Copy { walletName:'keepkey', rpcUrl:RPC_URL,// url to connect to an RPC endpoint (ie infura) }
```

- cobovault - (desktop & mobile support)
-

```
Copy { walletName:'cobovault', rpcUrl:RPC_URL,// url to connect to an RPC endpoint (ie infura) appName:APP_NAME// app name that is displayed on device }
```

- keystone - (desktop & mobile support)
-

```
Copy { walletName:'keystone', rpcUrl:RPC_URL,// url to connect to an RPC endpoint (ie infura) appName:APP_NAME// app name that is displayed on device }
```

- walletLink -``
- Coinbase Wallet
- ``for desktop (desktop only support)
-

```
Copy { walletName:'walletLink', rpcUrl:RPC_URL,// url to connect to an RPC endpoint (ie infura) appName:, // string to label your app appLogoUrl: // string url of your app logo [OPTIONAL] }
```

- imToken - (mobile only support)
-

```
Copy { walletName:'imToken', rpcUrl:RPC_URL// url to connect to an RPC endpoint (ie infura)[OPTIONAL] }
```

- meetone - (mobile only support)
-

```
Copy { walletName:'meetone'}
```

- mykey - (mobile only support)
-

```
Copy { walletName:'mykey', rpcUrl:RPC_URL// url to connect to an RPC endpoint (ie infura)[OPTIONAL] }
```

- huobiwallet - (mobile only support)
-

```
Copy { walletName:'huobiwallet', rpcUrl:RPC_URL// url to connect to an RPC endpoint (ie infura)[OPTIONAL] }
```

- hyperpay - (mobile only support)
-

```
Copy { walletName:'hyperpay'}
```

- wallet.io - (mobile only support)
-

```
Copy { walletName:"wallet.io",rpcUrl:RPC_URL}// url to connect to an RPC endpoint (ie infura)[OPTIONAL]
```

- atoken - (mobile only support)
-

```
Copy { walletName:"atoken"}
```

- frame - (mobile only support)
-

```
Copy { walletName:"frame"}
```

- ownbit - (mobile only support)
-

```
Copy { walletName:"ownbit"}
```

- alphawallet - (mobile only support)
-

```
Copy { walletName:"alphawallet"}
```

- gnosis - (desktop only support)
-

Gnosis wallet is unique as it designed to be used within the Gnosis Safe Apps interface. If the user opens your dApp within the Safe Apps interface, Gnosis wallet will be automatically selected and the select wallet modal will not be shown. This is to ensure that the user doesn't mistakenly use their wallet instead of their Safe Multisig wallet for the transaction.

```
Copy { walletName:"gnosis"}
```

- xdefi - (desktop & mobile support)
-

```
Copy { walletName:"xdefi"}
```

- bitpie - (mobile only support)
-

```
Copy { walletName:"bitpie"}
```

- liquality - (desktop only support)
-

```
Copy { walletName:"liquality"}
```

- binance - (desktop only support)
-

```
```

Copy { walletName:'binance'}

```

- mewwallet - (mobile only support)
-

```

Copy { walletName:'mewwallet', rpcUrl:RPC_URL,// url to connect to an RPC endpoint (ie infura) }

```

- tally - (desktop only support)
-

```

Copy { walletName: 'tally' }

```

- blankwallet - (desktop only support)
-

```

Copy { walletName: 'blankwallet' }

```

- mathwallet - (desktop & mobile support)
-

```

Copy { walletName: 'mathwallet' }

```

- 1inch - (mobile only support)
-

```

Copy { walletName: '1inch' }

```

- Did - (mobile only support)
-

```

Copy { walletName: 'did' }

```

- ronin - (desktop only support)
-

Ronin now only works on Axie's domains and will open to be used by other domains when the Ronin chain is readyUpdates From Ronin Dev

Due to the Ronin extension restriction, please add127.0.0.1 dev.skymavis.one in your hosts file/etc/hosts and accessdev.skymavis.oneinstead oflocalhost: .

```

Copy { walletName: 'ronin' }

```

Hardware Wallet Custom Networks

Onboard now supports passing in custom network info for hardware wallets allowing users to use their hardware wallets with non-Ethereum networks such as Energy Web Chain. ThecustomNetwork object is passed to theCommon constructor of the@ethereumjs/common library. This is then used to build the transaction that will be signed. Please refer to these instructions for more information. Note: Currently only Trezor and Ledger support this functionality/

Example

```

Copy const onboard = Onboard({ networkId: 1234 //... other options walletSelect: { wallets:[{ walletName: 'ledger', rpcUrl: RPC_URL, customNetwork: { networkId: number genesis: GenesisBlock hardforks: Hardfork[] bootstrapNodes: BootstrapNode[] } },] } })

```

You can view an example configuration[here](here) .

Wallet Check Modules

Initialization:

To initialize the built in wallet check modules, an array of wallet check initialization objects needs to be passed to thewalletCheck parameter of the Onboard initialization object. The order of the array is the order that the checks will be performed.

Each wallet check initialization object has additional optional customization parameters:

- heading
- -String
- Will over ride the heading displayed in the wallet check modal.
- description
- -String
- Will over ride the description displayed in the wallet check modal.
- icon
- -String
- Ansvg
- string that will over ride the icon displayed in the wallet check modal.
- button
- -{ text: String, onclick: () => void}
- An object that defines an extra button's text and onclick function.
- html
- -String
- A html string that will be rendered in the modal as html markup underneath the description.
- 

```

Copy importOnboardfrom'bnc-onboard'

constwalletChecks=[{ checkName:'derivationPath'}, { checkName:'accounts'}, { checkName:'connect'}, { checkName:'network'}, { checkName:'balance',minimumBalance:'1000000'}]

constonboard=Onboard({ //... other options walletCheck:walletChecks })

```

The following wallet check modules are included in Onboard:

- connect
- : Checks that the Dapp has access to the users' accounts and fires the connect function if the selected wallet has one to prompt the user to allow access.
- 

```

Copy { checkName:'connect'}

```

- derivationPath
- : Provides a UI to allow a user to select or input the derivation path they would like to use with their hardware wallet. Note: will only display if the connected wallet is of type "hardware".Note:
- html
- andbutton
- customization parameters won't work for this module as it needs them itself for correct operation.
- 

```

Copy { checkName:'derivationPath'}

```

- accounts
- : Provides a UI to allow a user to select and load accounts on their hardware wallet.Note:
- will only display if the connected wallet is of type "hardware".Note:
- html
- andbutton
- customization parameters won't work for this module as it needs them itself for correct operation.
- 

```

Copy { checkName:'accounts'}

```

- network
```

- : Checks that the users' wallet is connected to the network that the modules were initialized with.
-

```

Copy { checkName:'network'}

```

- balance
- : Checks that the users' account has the minimum balance as defined when initialized.
-

```

Copy { checkName:'balance', minimumBalance:'1000000'// The minimum balance in wei required [String] }

```

In order for Onboard to efficiently detect wallet balance changes, adappId is required when initializing Onboard. This feature is subject to ourAPI Rate Limits .

Creating Custom Modules

Onboard has been built so that custom wallet and wallet check modules can be plugged in to allow for the breadth of different wallets and on-boarding flows that are needed across the web3 ecosystem.

Custom Wallet Modules

A wallet module is an abstraction that allows interoperability between wallets within Onboard.

A wallet module has the following properties:

name -[string][required]

The name of the wallet.

wallet -[function][required]

A function that is called when this wallet is selected. It is called with ahelpers object that includes some helper functions that make it easier to create the return object. Thehelpers object contains the following methods:

- getProviderName
-
  - A function that takes a provider as an argument and will return the name of the most common providers.
- createLegacyProviderInterface
-
  - A function that takes the provider as an argument and returns awalletInterface
- . Legacy refers to a provider thathas not
- implementedEIP-1102
- .
- createModernProviderInterface
-
  - A function that takes the provider as an argument and returns awalletInterface
- . Modern refers to a provider thathas
- implementedEIP-1102
- . This will work for most providers and is a quick way to create an interface.
- BigNumber
-
  - Abignumber.js
- function for number calculations.
- getNetwork
-
  - A function that takes a provider and will make aJSON-RPC
- call to get the network.
- getAddress
-
  - A function that takes a provider and will make aJSON-RPC
- call to get the address.
- getBalance
-
  - A function that takes a provider and will make aJSON-RPC
- call to get the balance.
-

Thewallet function must return aPromise that resolves to an object that has the following properties:

provider - the JavaScript provider for this wallet.[required]

instance - forSDK wallets instances that are separate from theprovider .[optional]

interface - awalletInterface ornull if the user's current provider is not this wallet(for extension and mobile wallets) .[required] walletInterface

An abstraction built on top of anEthereum JavaScript provider allowing interoperability with wallets and providers of any architecture. A validwalletInterface implements the following methods and attributes:

name - The name of the wallet.[string][required]

connect - A function that requests account access and returns aPromise that resolves or rejects once the user confirms or rejects the request. [function][optional]

disconnect - A function to clean up / destroy any instances (mostly forSDK wallets). Will be called when a new wallet has been selected.[function][optional]

loading - A function that returns aPromise that resolves when the wallet has finished loading. This is to show a loading spinner as the wallet popup is loading (mostly forSDK wallets).[function][optional]

address - AstateSyncer object (see below for details). Theget function on thisstateSyncer must return a string which is the wallet address.

network - AstateSyncer object (see below for details). Theget function on thisstateSyncer must return a number which is the wallet network id.

balance - AstateSyncer object (see below for details). Theget function on thisstateSyncer must return a string which is the wallet address balance.

stateSyncer

An object that has either aget function or aonChange function and is used to keep user state in sync. TheonChange method is preferred as it doesn't require background polling to keep in sync.

Theget function returns a promise that resolves with a value.

TheonChange function takes asetState function as an argument and is used to set the state. Often it is needed to use thesetState function to set the initial state and then register it to be called every time it changes.

iconSrc -[string][optional]

A source for the wallet logo icon.

iconSrcSet -[string][optional]

A source for the hi-res wallet logo icon.

svg - ****[string][optional]

Asvg element string for the wallet logo icon.

link -[string][optional]

A link to install this wallet (extension and injected provider wallets).

installMessage -[function][optional]

A function that will be called when a user has selected this wallet, but thewallet function did not return a walletinterface . This is generally only needed if this wallet injects the provider via an extension or mobile wallet and the user doesn't have the extension installed or is on the wrong mobile browser. The function will be called with an object that has two properties:currentWallet (the name of the wallet that has been detected by Onboard) andselectedWallet (the name of the selected wallet), and must return a string (that can includehtml) that instructs the user to install this wallet.

desktop -[boolean][optional]

Whether the wallet is a desktop wallet or not.

mobile -[boolean][optional]

Whether the wallet is a mobile wallet or not.

preferred -[boolean][optional

Whether this wallet will be displayed as a preferred wallet.

Example

```

Copy importOnboardfrom'bnc-onboard' importSomeSDKWalletfrom'some-sdk-wallet'

constcustomExtensionWalletLogo=` <svg height="40" viewBox="0 0 40 40" width="40" xmlns="http://www.w3.org/2000/svg"

    `

constcustomSDKWalletLogo=` <svg height="40" viewBox="0 0 40 40" width="40" xmlns="http://www.w3.org/2000/svg"

    `

// create custom wallet constcustomExtensionWallet={ name:'Extension Wallet', svg:customExtensionWalletLogo, wallet:asynchelpers=>{ const{createModernProviderInterface}=helpers constprovider=window.ethereum
```

constcorrectWallet=window.ethereum.currentProvider.isExtensionWallet

return{ provider, interface:correctWallet?createModernProviderInterface(provider):null } }, link:'https://some-extension-wallet.io', installMessage:wallets=>{ const{currentWallet,selectedWallet}=wallets if(currentWallet) { returnYou have{currentWallet}installed already but if you would prefer to use{selectedWallet}instead, then click below to install. }

returnYou will need to install{selectedWallet}to continue. Click below to install. }, desktop:true }

// create custom wallet constcustomSDKWallet={ name:'SDK Wallet', svg:customSDKWalletLogo, wallet:asynchelpers=>{ const{createModernProviderInterface}=helpers constinstance=newSomeSDKWallet({ apiKey:'sd3d3fwd'}) constprovider=instance.getProvider()

return{ provider, interface:createModernProviderInterface(provider), instance } }, desktop:true, mobile:true }

constdefaultWallets=[{ walletName:'metamask'}, { walletName:'coinbase'}]

// initialize onboard constonboard=Onboard({ //... other options walletSelect:{ wallets:[...defaultWallets,customExtensionWallet,customSDKWallet] }, //... other options } })

```

Custom Wallet Check Modules

You can create custom onboarding flows by stringing together a sequence of wallet check modules that check that the users' state is correct before moving forward. A wallet check module is a function that gets called with an object which contains the users' current state and either returns a result ofundefined if that check has passed or amodalObject if it failed. A wallet check module can also return aPromise that resolves to the check result, allowing asynchronous actions and checks to be a part of your onboarding flow.

A wallet check module will be called with a state object with the following parameters and functions:

address - The users' current address.[string]

network - The users' current network id.[number]

balance - The users' current account balance inwei .[string]

appNetworkId - ThenetworkId that the Dapp is configured to work with.[number]

wallet - The users' wallet object containing the properties:provider ,interface ,connect ,name ,loading andinstance .[object]

mobileDevice - Whether the user is on a mobile device or not.[boolean]

BigNumber - Abignumber.js function for number calculations.[function]

walletSelect - Call this to switch wallets, will close wallet check dialog and open wallet select dialog.[function]

exit - Call to exit wallet check dialog.[function]

modalObject

The module function then needs to evaluate a condition and then return a result ofundefined if the condition has been met or amodalObject if not. ThemodalObject is an object with the following properties:

heading - The heading for the wallet check modal.[string][required]

description - The description for the wallet check modal.[string][required]

eventCode - An event code that is used for your back-end analytics dashboard.[string][required]

**button ** - ** A object with anonclick property to register a function to run when the button is clicked, and atext property for the button text[object][optional]

html - A html string that will be rendered in the modal as html markup underneath the description[string][optional]

icon - A icon to be displayed next to the heading.[string][optional]

action - A function to run upon showing the modal that returns a Promise that resolves when finished or rejects with an error.[function][optional]

Example

For a user to interact with your Dapp, they may require a token balance (ieDai ). You could build a module that checks that a user has the required token balance to continue:

```

Copy importOnboardfrom'bnc-onboard' importethersfrom'ethers' importerc20from'./erc20'

functiontokenBalance({ tokenAddress,minimumBalance,tokenName }) { letethersProvider; lettokenContract;

returnasyncstateAndHelpers=>{ const{ wallet: {provider}, address, BigNumber }=stateAndHelpers;

if(!tokenContract) { ethersProvider=newethers.providers.Web3Provider(provider); tokenContract=newethers.Contract(tokenAddress,erc20,ethersProvider); }

consttokenDecimals=awaittokenContract.decimals(); constdivideBy=newBigNumber(10).pow(tokenDecimals); consttokenBalanceResult=awaittokenContract .balanceOf(address) .then(res=>res.toString()); consttokenBalance=newBigNumber(tokenBalanceResult).div(divideBy);

if(tokenBalance.lt(minimumBalance)) { return{ heading:Get Some{tokenName}, description:You need to have at least{minimumBalance}{tokenName}to interact with this Dapp. Send some more{tokenName}to this address or switch to another address that has a higher{tokenName}balance., eventCode:"tokenBalance", icon:` <svg height="18" viewBox="0 0 429 695" width="18" xmlns="http://www.w3.org/2000/svg"

<g fill="currentColor" fill-rule="evenodd"

` }; } }; }

constdefaultWalletChecks=[{ checkName:'connect'}, { checkName:'network'}, { checkName:'balance',minimumBalance:'100000'}]

consttokenBalanceCheck=tokenBalance({tokenAddress:'0x6b175474e89094c44da98b954eedeac495271d0f',tokenName:'Dai',minimumBalance:5})

constonboard=Onboard({ //... other options walletCheck:[...defaultWalletChecks,tokenBalanceCheck] })

```

Caching Wallet Selection

You may want to cache the wallet a user has selected previously so that you can automatically select it for them next time they visit your app for a silky smooth user experience. The first step is to store the wallet name in local storage when the user selects a wallet. You can do this by adding some code to your wallet subscription function inside the onboard config:

```

Copy constconfig={ //... other config options subscriptions:{ wallet:wallet=>{ // store the selected wallet name to be retrieved next time the app loads window.localStorage.setItem('selectedWallet',wallet.name) } } }

constonboard=Onboard(config)

```

Then you can add code that runs when your app first load, that will retrieve theselectedWallet value fromlocalStorage and then callwalletSelect with that value. Onboard will then automatically select it for the user if that wallet is available:

```

Copy functiononLoad() { // ... initialize onboard

// get the selectedWallet value from local storage constpreviouslySelectedWallet=window.localStorage.getItem('selectedWallet')

// call wallet select with that value if it exists if(previouslySelectedWallet!=null) { awaitonboard.walletSelect(previouslySelectedWallet) } }

```

Adding Networks/Chains

Certain wallets give the option to add networks or chains from predefined configurations. This example will use MetaMask.

Once connected and a walletprovider exposed thewallet_addEthereumChain method can be called with specific configurations to setup that ethereum based chain within the wallet.

```

Copy const addNetwork = (chainId) => { if (!provider) return; provider .request({ method: 'wallet_addEthereumChain', params: [ethereumNetworks[chainId]], }) .then(resolve) .catch(reject); }

export const ethereumNetworks = { 56: { chainId:0x{parseInt(56, 10).toString(16)}, chainName: 'BSC Mainnet', nativeCurrency: { name: 'Binance Coin', symbol: 'BNB', decimals: 18, }, rpcUrls: ['https://bsc-dataseed.binance.org'], blockExplorerUrls: ['https://bscscan.com/'], }, 43114: { chainId: 0x{parseInt(43114, 10).toString(16)}, chainName: 'Avalanche C-Chain', nativeCurrency: { name: 'AVAX', symbol: 'AVAX', decimals: 18, }, rpcUrls: ['https://api.avax.network/ext/bc/C/rpc'], blockExplorerUrls: ['https://snowtrace.io/'], }, 137: { chainId: 0x{parseInt(137, 10).toString(16)}, chainName: 'Polygon Mainnet', nativeCurrency: { name: 'MATIC', symbol: 'MATIC', decimals: 18, }, rpcUrls: ['https://polygon-rpc.com'], blockExplorerUrls: ['https://polygonscan.com/'], }, ... };

```

Once the network is setup within the walletwallet_switchEthereumChain can then be called to switch to chains that have been setup.

More info around these method calls can be foundhere

Customizing CSS

To gain the required specificity to override the default styles, you need to use the class name.bn-onboard-custom plus the class name that you would like to override in yourCSS file. For example, if you would like to customize thefont-family for the onboard modals, then you would target the top level element that has the class.bn-onboard-modal in yourCSS file:

```

Copy .bn-onboard-custom.bn-onboard-modal{ font-family:Roboto; }

```

Below is a list of the main classes that you can target to customize theCSS :

.bn-onboard-modal - The full page modal container.
```

.bn-onboard-modal-content - The container for the modal content.

.bn-onboard-modal-header - The header of the modal

.bn-onboard-modal-content-header-heading - The heading within the header.

.bn-onboard-select-description - The wallet select modal description.

.bn-onboard-modal-select-wallets - The unordered list of wallets.

.bn-onboard-icon-button - The wallet button.

.bn-onboard-modal-selected-wallet - Selected wallet modal content.

.bn-onboard-prepare-button - All modal buttons

Thez-index property for all Onboard modals was removed in version1.9.6 to prevent interference with wallet modals. If you would like to restore thez-index setting from previous versions then add the following to your CSS:.bn-onboard-custom.bn-onboard-modal { z-index: 99; }

Get Started Today

Sign up for a free Blocknative Account athttps://explorer.blocknative.com/account with your work email address.

If you have any questions, connect with the team on ourdiscord

React Demo

To see an Onboard integration in a simple React app you can check out ourreact-demo repo on GitHub:https://github.com/blocknative/react-demo We have thereactdemo codebase deployed so that you try Onboard from an end user perspective:https://reactdemo.blocknative.com/

Rollup Builds

For projects using Webpack for bundling, Onboard.js will work out of the box. If you are using Rollup as your bundler you will need to install some extra plugins and add some configurations to yourrollup.config.js file due to Rollup not including thenode built-in functions automatically. The script tag in yourindex.html will also need to be modified to allow for code-splitting.

Ledger andTrezor wallets do not currently work in Rollup due to the lack of maintained support for node built-ins. To get your build to work you need tonot include wallet initialization objects for those wallets in your Onboard config. The plugins you will need to install asdevDependencies are:

- @rollup/plugin-json
- @rollup/plugin-commonjs
- @rollup/plugin-node-resolve
- rollup-plugin-node-polyfills
-

The basicrollup.config.js configuration looks like this:

```

Copy importresolvefrom'@rollup/plugin-node-resolve' importcommonjsfrom'@rollup/plugin-commonjs' importjsonfrom'@rollup/plugin-json' importnodePolyfillsfrom'rollup-plugin-node-polyfills'

exportdefault{ input:'src/main.js', output:{ format:'esm',// needs to be esm format as Onboard.js contains code-splitting dir:'public/esm/' }, plugins:[ json(), resolve({ browser:true, preferBuiltins:true }), commonjs({ namedExports:{ "u2f-api":["sign","isSupported"] } }), nodePolyfills() ] }

```

In yourindex.html you need to addtype="module" to the script tag that imports your bundle to enable code-splitting:

```

Copy

```

On this page * Quickstart * Create a Blocknative Account * Create your API Key * Install the widget using npm * Initialize the Library * Onboard User * And you are live! * Screencasts * Example Code * Initialization * Options * Initialize Onboard * Customizing 'unknown' Network Name * Local Networks * API * walletSelect * walletCheck * walletReset * accountSelect * getState * config * Built-in Modules * Wallet Modules * Hardware Wallet Custom Networks * Wallet Check Modules * Creating Custom Modules * Custom Wallet Modules * Custom Wallet Check Modules * Caching Wallet Selection * Adding Networks/Chains * Customizing CSS * Get Started Today * React Demo * Rollup Builds

Was this helpful?