Solana Private Key Provider for PnP Web SDKs

@web3auth/solana-provider

â

The Solana Blockchain Provider is basically a wrapper around theSolana JSON RPC API making it easier to interact with the Solana Blockchain.

In this section we'll explore more about how you can use this provider with our SDKs.

Installationâ

@web3auth/solana-provider

â

npm install --save @web3auth/solana-provider

Initialisationâ

Import theSolanaPrivateKeyProvider class from@web3auth/solana-provider .

import

{

SolanaPrivateKeyProvider

}

from

"@web3auth/solana-provider" ;

Assign theSolanaPrivateKeyProvider

class to a variableâ

After creating your Web3Auth instance, you need to initialize the Torus Wallet UI Plugin and add it to a class for further usage.

const privateKeyProvider =

new

SolanaPrivateKeyProvider ({

config :

SolanaPrivKeyProviderConfig

}) ; This constructor takes an object with a config ofSolanaPrivKeyProviderConfig as input.

Argumentsâ

SolanaPrivKeyProviderConfig

export

interface

SolanaPrivKeyProviderConfig

extends

BaseProviderConfig

{ chainConfig :

Omit < CustomChainConfig ,

"chainNamespace"

; } export

type

CustomChainConfig

=

{ chainNamespace :

ChainNamespaceType ; /* * *The chain id of the chain*/ chainId :

string ; /* * *RPC target Url for the chain*/ rpcTarget :

string ; /* * *web socket target Url for the chain*/ wsTarget ? :

string ; /* * *Display Name for the chain*/ displayName :

string ; /* * *Url of the block explorer*/ blockExplorer :

string ; /* * *Default currency ticker of the network (e.g: ETH)*/ ticker :

string ; /* * *Name for currency ticker (e.g:Ethereum)* / tickerName :

string ; /* * *Number of decimals for the currency ticker (e.g: 18)*/ decimals ? :

number ; } ; export

interface

BaseProviderConfig

extends

BaseConfig

{ chainConfig :

Partial < CustomChainConfig

; networks ? :

Record < string ,

CustomChainConfig

; skipLookupNetwork ? :

boolean ; } export

interface

BaseConfig

{ /* * *Determines if this controller is enabled* / disabled ? :

boolean ; }

Getting thechainConfig

â

- Mainnet
- Testnet

const chainConfig :

{ chainNamespace :

CHAIN_NAMESPACES . SOLANA ; chainId :

"0x1" ;

// Please use 0x1 for Mainnet, 0x2 for Testnet, 0x3 for Devnet rpcTarget :

"https://rpc.ankr.com/solana" ; displayName :

"Solana Mainnet" ; blockExplorerUrl :

"https://explorer.solana.com" ; ticker :

"SOL" ; tickerName :

"Solana" ; logo :

"https://images.toruswallet.io/solana.svg" ; } ; const chainConfig :

{ chainNamespace :

CHAIN_NAMESPACES . SOLANA ; chainId :

"0x2" ;

// Please use 0x1 for Mainnet, 0x2 for Testnet, 0x3 for Devnet rpcTarget :

"https://api.testnet.solana.com" ; displayName :

"Solana Testnet" ; blockExplorerUrl :

"https://explorer.solana.com" ; ticker :

"SOL" ; tickerName :

"Solana" ; logo :

"https://images.toruswallet.io/solana.svg" ; } ;

Initializing and instantiating the Web3Auth SDKâ

- PnP Modal SDK
- PnP NoModal SDK
- CoreKit SFA Web SDK

import

{

Web3Auth

}

from

"@web3auth/modal" ; import

{

SolanaPrivateKeyProvider

}

from

"@web3auth/solana-provider" ; import

{

WEB3AUTH_NETWORK

}

from

```
"@web3auth/base" ;
const privateKeyProvider =
new
SolanaPrivateKeyProvider ( { config :
{ chainConfig : chainConfig } , } ) ;
const web3auth =
new
Web3Auth ( { // Get it from Web3Auth Dashboard clientId , web3AuthNetwork :
WEB3AUTH_NETWORK . SAPPHIRE_MAINNET , privateKeyProvider : privateKeyProvider , } ) ; import
{
Web3AuthNoModal
}
from
"@web3auth/no-modal" ; import
{
AuthAdapter
}
from
"@web3auth/auth-adapter" ; import
{
SolanaPrivateKeyProvider
}
from
"@web3auth/solana-provider" ; import
{
WEB3AUTH_NETWORK
}
from
"@web3auth/base" ;
const privateKeyProvider =
new
SolanaPrivateKeyProvider ( { config :
{ chainConfig } , } ) ;
const web3auth =
new
Web3AuthNoModal ( { clientId ,
// Get it from Web3Auth Dashboard web3AuthNetwork :
```

WEB3AUTH_NETWORK . SAPPHIRE_MAINNET , privateKeyProvider , }) ;

const authAdapter =

new

AuthAdapter ({ privateKeyProvider }) ; web3auth . configureAdapter (authAdapter) ; import

{

Web3Auth

}

from

"@web3auth/single-factor-auth" ; import

{

SolanaPrivateKeyProvider

}

from

"@web3auth/solana-provider" ;

const privateKeyProvider =

new

SolanaPrivateKeyProvider ({ config :

{ chainConfig } , }) ;

const web3auth =

new

Web3Auth ({ clientId ,

// Get your Client ID from Web3Auth Dashboard web3AuthNetwork :

"sapphire_mainnet" , privateKeyProvider , }) ;

Using the providerâ

On connection, you can useweb3auth.provider as a solana provider with@web3auth/solana-provider along with@solana/web3.js library.

import

{

SolanaWallet

}

from

"@web3auth/solana-provider" ;

const solanaWallet =

new

SolanaWallet (provider) ; Once you have setup the provider, you can use the standard functions in thesolana/web3.js library to get user's account, perform transaction, sign a message etc. Here we have listed a few examples to help you get started there:

info All the RPC methods which are available by default on Solana Blockchain are also available on the Solana Provider. Although, for the case ofphantom adapter they are not available.

You can refer to standard RPC calls for Solana here tip Please refer to our Solana Connect Blockchain Reference for more information.

Examplesâ

Integrate PnP Modal SDK with Solana Blockchain SAMPLE APP Use Solana with Plug and Play Modal SDK Source Code Guide pnp web @web3auth/modal javascript solana ed25519 ### Integrate PnP No Modal SDK with Solana Blockchain SAMPLE APP Use Solana with Plug and Play No Modal SDK Source Code Guide pnp web @web3auth/no-modal javascript solana ed25519 Edit this page Previous Account Abstraction Provider Next XRPL Provider