overview)

Was this helpful? Export as PDF

Migrate from v2 to v3

Migration guide for upgrading LI.FI SDK v2 to v3

Overview

LI.FI SDK v3 has undergone a major update, improving compatibility with popular libraries likeViem and adding new features, including support for multiple ecosystems, starting withSolana . Consequently, as detailed in this guide, you need to be aware of some breaking changes and deprecations. We also recommend reviewing the updated documentation for additional new features not covered here.

To get started, install the latest version of LI.FI SDK.

yarn pnpm bun npm ```

Copy yarnadd@lifi/sdk

Copy pnpmadd@lifi/sdk

Copy bunadd@lifi/sdk

Copy npminstall@lifi/sdk

```

Configuration

We have made significant changes to how the LI.FI SDK is configured. It is no longer class-based, so you don't need to create, maintain, and share a LiFi class instance to use SDK functionality. Instead, it is now function-based, allowing you to configure it once and update the configuration from any place without needing to maintain or share the configuration object's reference. You can simply import the necessary functions from the package wherever needed. Read moreConfigure SDK

In this example, you can call thegetQuote function from anywhere using SDK v3, whereas with SDK v2, you had to share a createdLiFi class instance and callgetQuote as a method of that class.

```

Copy // SDK v2 import{ ChainId,LiFi }from'@lifi/sdk'

constlifi=newLiFi({ integrator:'Your dApp/company name' })

constquote=awaitlifi.getQuote({ fromAddress:'0xd8dA6BF26964aF9D7eEd9e03E53415D37aA96045', fromChain:ChainId.ARB, toChain:ChainId.OPT, fromToken:'0x0000000000000000000000000000000000000000', toToken:'0x0000000000000000000000000000000000000000', fromAmount:'1000000000000000000', })

// SDK v3 import{ ChainId,createConfig,getQuote }from'@lifi/sdk'

createConfig({ integrator:'Your dApp/company name', })

constquote=awaitgetQuote({ fromAddress:'0xd8dA6BF26964aF9D7eEd9e03E53415D37aA96045', fromChain:ChainId.ARB, toChain:ChainId.OPT, fromToken:'0x0000000000000000000000000000000000000000', toToken:'0x0000000000000000000000000000000000000000', fromAmount:'1000000000000000000', })

```

Due to these changes, the configuration-related functions from theLiFi interface are no longer needed. The previous
```

interface included:

```
Copy interfaceLiFi{ getConfig():Config getConfigAsync():Promise getRpcProvider(chainId:number,archive:boolean):Promise setConfig(configUpdate:ConfigUpdate):Config // ... }
```

Now, you can import the configuration object directly and make any necessary changes to it. This simplifies the process and improves code maintainability. Read moreUpdate SDK configuration .

```
Copy // SDK v2 import{ ChainId,LiFi }from'@lifi/sdk'

constlifi=newLiFi({ integrator:'Your dApp/company name' })

lifi.setConfig({ integrator:'Your dApp/company name' })

// SDK v3 import{ config }from'@lifi/sdk';

config.set({ integrator:'Your dApp/company name', });
```

Renamed methods

All methods previously supported by the class-based approach are now individual functions that you can import directly from the@lifi/sdk package. Some of these methods have been renamed to better reflect their functionality or to have a more concise name.

- getContractCallQuote
- ->getContractCallsQuote
- getTokenApproval
- ->getTokenAllowance
- bulkGetTokenApproval
- ->getTokenAllowanceMulticall
- approveToken
- ->setTokenAllowance
- moveExecutionToBackground
- ->updateRouteExecution

SeeRequest contract call Quote ,Manage route execution andToken Management for more details.

Moving from Ethers.js to Viem

Since the first version of the LI.FI SDK, we have relied on theEthers.js library for all EVM-related interactions. However, as the industry evolves,Viem is gaining wide adoption and starting to replaceEthers.js .

To ensure our product remains robust and future-proof, we have decided to transition our entire stack to a more reliable solution -Viem (Wagmi for the Widget).

Please see theEthers v5 → viem Migration Guide for more details. Among other notable changes, this transition also replaces all BigNumber utilities we previously used in v2 with the nativebigint primitive.

Multiple ecosystem support

LI.FI SDK v3 now supports two ecosystems — EVM and Solana — with more on the way. To accommodate these new ecosystems, we have introduced the concept of ecosystem providers in SDK v3. We extracted all EVM-related functionality from SDK v2 and integrated it into an EVM provider. Additionally, a Solana provider has been added to enable Solana-related functionality across all SDK functions.

These ecosystem providers are designed with modularity in mind and are fully tree-shakable, ensuring that they do not add unnecessary weight to your bundle if not used.

Previously, it was necessary to pass anEthers.js Signer object toexecuteRoute and other functions for EVM interactions. With the introduction of ecosystem providers, you only need to configure them once when setting up the SDK. After that, you can useexecuteRoute and other functions without passing any additional options. The SDK will automatically determine the appropriate ecosystem provider and notify you if no suitable provider is configured.

Read moreIntroduction to SDK Ecosystem Providers .

Examples

Check out our complete examples in the SDK repository , and feel free to file an issue if you encounter any problems.

Changelog

For a detailed view of all the changes, please see the CHANGELOG .

Install LI.FI SDK Last updated4 months ago