Migrating to LI.FI SDK v2

Migration guide of upgrading LI.FI's SDK v1 to v2 This migration guide is a quick reference to the upgrades/fixes in LI.FI's SDK v2. All the breaking changes, deprecations, improvements, etc, are listed below. We advise going through all the changes and updating the SDK implementation in your project accordingly.

Breaking Changes

Deprecation ofgetByChainID method

The methodgetByChainID is no longer available in the SDK, useChainsService instead.

```

Copy - const chainResponse = getChainById(Number(chainId));

- import ChainsService from './ChainsService' +
- const chainService = ChainsService.getInshttps://meet.google.com/mda-wkwc-jmj?authuser=user%40li.financetance()
- const chainResponse = await chainService.getChainById(fromChainId)

```

RenameupdateCallback method in settings

The methodupdateCallback is renamed toupdateRouteHook

```

Copy import { LiFi } from "@lifi/sdk"

const lifi = new LiFi(lifiConfig);

const routeUpdateCallback = () => { ... }

const executionSettings: ExecutionSettings = { - updateCallback: routeUpdateCallback, + updateRouteHook: routeUpdateCallback }

const response = await lifi.execute(signer, route, executionSettings);

```

RenameacceptSlippageUpdateHook callback method in settings

The methodacceptSlippageUpdatehook is renamed toacceptExchangeRateUpdateHook .

```

Copy const exchangeRateUpdateCallback = () => { ... }

const executionSettings: InternalExecutionSettings = { - acceptSlippageUpdateHook: exchangeRateUpdateCallback, + acceptExchangeRateUpdateHook: exchangeRateUpdateCallback, }

```

Add integrator during SDK initialisation

During the initialisation of the SDK, you should pass theintegrator as part of the configuration object.

The configuration to instantiate LiFi is nowmandatory .

```

Copy import{ Lifi }from'@lifi/sdk'

// Mandatory to pass the configuration object constlifiConfig={ integrator:"your-integrator-name" }

constlifi=newLiFi(lifiConfig)

```

By providing theintegrator during the initialisation of the SDK, it'll be included as request headers for all requests made by the SDK which would later be used for tracking and identification purposes on the server side.

Returning Promise instance of Route

Any route interaction to SDK would now return the initial Promise instance of Route created during the first execution of Route. In order to get more details about the Route, developer would need to resolve the Promise to access the information.

⚒ Improvements

AddconvertQuoteToRoute helper function to execute single-step quotes

For single-step executions a new helper function,convertQuoteToRoute is added to the SDK. This function takes aStep object as input and returns aRoute object. It is useful for converting quotes to routes, which is necessary for executing transactions.

```

Copy import{ convertQuoteToRoute }from'@lifi/sdk'; import{ LiFi }from'@lifi/sdk';

constlifi=newLiFi(lifiConfig);

constquoteWithSingleStep=awaitlifi.getQuote(quoteRequest);

// Use the convertQuoteToRoute function to convert the Step to a Route
constroute:Route=convertQuoteToRoute(quoteWithSingleStep);

// Now, you can use the route object for executing transactions constresponse=awaitlifi.executeRoute(route);

```

AddgetConnections method to LI.FI SDK

A new methodgetConnections is added to the SDK, which retrieves available connections for swapping or bridging tokens. This function takes aConnectionsRequest object as a parameter and returns aConnectionsResponse object.

```

Copy import{ LiFi }from'@lifi/sdk';

constlifi=newLiFi(lifiConfig);

constrequest:ConnectionsRequest={ fromChain:1, fromToken:'0x123456789abcdef', toChain:137,
toToken:'0x987654321fedcba', allowBridges:['connext','uniswap'] }

constconnections=awaitlifi.getConnections(request);

```

Add support to modify Transaction Request

A new callback is added that allows developers to customise the gas configuration for the transaction request before sending the transaction to the blockchain.

The callback can be set inExecutionSettings while interacting with the SDK.

```

Copy import{ LiFi }from'@lifi/sdk';

constlifi=newLiFi();

// define your logic for modifying the gas config of the transaction request
constupdateGasConfig=async(txRequest:TxRequest):Promise=>{ constcustomGasConfig=awaitcustomGas();
constupdatedTxRequest={ ...txRequest, ...customGasConfig }

returnupdatedTxRequest; }

// add callback function to modify the gas configurations, in settings constsettings:ExecutionSettings={ signer, route, settings:
{ updateTransactionRequestHook:updateGasConfig } }

constresponse=awaitlifi.executeRoute(signer,route,settings)
```

```

Only the modified gas config variable such asgasLimit ,gasPrice ,maxPriorityFeePerGas andmaxFeePerGas are picked up from it.

Optimise Chain Switching in Bridging Steps

The update refines theexecuteStep function's chain-switching logic, allowing users to leave the source chain once the transaction is signed. Users now only need to switch to the destination chain if a swap is required, streamlining the bridging process and improving the user experience.

CHANGELOG

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

Last updated29 days ago On this page * Breaking Changes * Deprecation of getByChainID method * Rename updateCallback method in settings * Rename acceptSlippageUpdateHook callback method in settings * Add integrator during SDK initialisation * Returning Promise instance of Route * ⚒ Improvements * Add convertQuoteToRoute helper function to execute single-step quotes * Add getConnections method to LI.FI SDK * Add support to modify Transaction Request * Optimise Chain Switching in Bridging Steps * CHANGELOG

Was this helpful? Export as PDF
```