4

Execute a Route (Transfer)

We allow you to execute any on-chain or cross-chain swap and bridging process and a combination of both. After fetching routes , you can execute one usingexecuteRoute as described here.

```

Copy asyncfunctionexecuteRoute( signer:Signer, route:Route, settings?:ExecutionSettings ):Promise

interfaceExecutionSettings{ acceptExchangeRateUpdateHook?:AcceptExchangeRateUpdateHook
switchChainHook?:SwitchChainHook updateRouteHook?:UpdateRouteHook
updateTransactionRequestHook?:TransactionRequestUpdateHook executeInBackground?:boolean
infiniteApproval?:boolean }

```

Executing a route requires a signer to send transactions to involved contracts. You can read more about signers in the official ethers documentation . Once you have the route and the signer, you callexecuteRoute.

```

Copy constroute=awaitlifi.executeRoute(signer,route)

```

Note that the function will return the executed route once the execution has been completed.

In addition to the first two parameters,executeRoute takes an optionalsettings object as a third parameter.

switchChainHook

page Handling Chain Switches

updateRouteHook

This function will be called every time the SDK performs an action on the route. You can use this callback to keep track of the execution status.

acceptExchangeRateUpdateHook

This function will be called every time the rate changes during swap/bridge and will provide old and new amount values to you. You should returntrue as a result to continue the execution. When you don't provide this hook or returnfalse - SDK throws an error (TransactionError: Exchange rate has changed!).

updateTransactionRequestHook

This hook is intended for advanced usage and you can use it to modify transaction or token approval requests e.g. updating gas price.

Controlling the execution of a route

moveExecutionToBackground

```

Copy functionmoveExecutionToBackground(route:Route):void

```

Once a route execution has started, it can be pushed "to the background" by calling this method. Once called, the execution will continue until it reaches a point where user interaction is required (i.e. signing a transaction). If such a point is reached, the execution will halt untilresumeRoute is called with the same route object. The execution will then pick up where it halted.

stopExecution

```

Copy functionstopExecution(route:Route):Route
```

```

This method immediately stops the execution of a given route. If a transaction has already been signed and sent by the user, it will be executed on-chain.

updateExecutionSettings

```

Copy functionupdateExecutionSettings( settings:ExecutionSettings, route:Route ):void

```

This function updates the execution settings of a route. Please see#executeroute for theExecutionSettings interface.

resumeRoute

```

Copy asyncfunctionresumeRoute( signer:Signer, route:Route, settings?:ExecutionSettings ):Promise

```

resumeRoute takes in the same parameters asexecuteRoute and will resume either a route that has been stopped or a route that has been moved to the background.

Example code snippet

Building on the code fromRequest a Route we can now choose a route and execute it.

```

Copy // getting routes constrouteOptions={...} constroutesRequest={...}

constresult=awaitlifi.getRoutes(routesRequest) constroutes=result.routes constchosenRoute=routes[0]

constupdateCallback=(updatedRoute:Route)=>{ console.log('Ping! Everytime a status update is made!') }

// executing a route constroute=awaitlifi.executeRoute(signer,chosenRoute,{ updateCallback })

``` Last updated2 months ago On this page *Controlling the execution of a route* Example code snippet

Was this helpful? Export as PDF
```