how-to-request-routes)

Request Routes/Quotes

Prior to executing any swap or bridging, you need to request the best route from our smart routing API. The LI.FI SDK provides functionality to request routes and quotes, as well as to execute them. This guide will walk you through the process of making a request usinggetRoutes andgetQuote functions.

How to request Routes

To get started, here is a simple example of how to request routes to bridge and swap 10 USDC on Arbitrum to the maximum amount of DAI on Optimism.

```

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

constroutesRequest:RoutesRequest={ fromChainId:42161,// Arbitrum toChainId:10,// Optimism fromTokenAddress:'0xaf88d065e77c8cC2239327C5EDb3A432268e5831',// USDC on Arbitrum toTokenAddress:'0xDA10009cBd5D07dd0CeCc66161FC93D7c9000da1',// DAI on Optimism fromAmount:'10000000',// 10 USDC };

constresult=awaitgetRoutes(routesRequest); constroutes=result.routes;

```

When you request routes, you receive an array of route objects containing the essential information to determine which route to take for a swap or bridging transfer. At this stage, transaction data is not included and must be requested separately. Read moreExecute Routes/Quotes .

Additionally, if you would like to receive just one best option that our smart routing API can offer, it might be better to request a quote usinggetQuote . SeeRequest a Quote section.

Routes request parameters

ThegetRoutes function expects aRoutesRequest object, which specifies a desiredany-to-any transfer and includes all the information needed to calculate the most efficient routes.

Parameters

Below are the parameters for theRoutesRequest interface along with their descriptions:

fromChainId (number, required)

The ID of the source chain (e.g., Ethereum mainnet is 1).

fromTokenAddress (string, required)

The contract address of the token on the source chain. Ensure this address corresponds to the specifiedfromChainId .

fromAmount (string, required)

The amount to be transferred from the source chain, specified in the smallest unit of the token (e.g., wei for ETH).

fromAddress (string, optional)

The address from which the tokens are being transferred.

toChainId (number, required)

The ID of the destination chain (e.g., Optimism is 10).

toTokenAddress (string, required)

The contract address of the token on the destination chain. Ensure this address corresponds to the specifiedtoChainId .

toAddress (string, optional)

The address to which the tokens will be sent on the destination chain once the transaction is completed.

fromAmountForGas (string, optional)

Part of theLI.Fuel . Allows receiving a part of the bridged tokens as gas on the destination chain. Specified in the smallest unit of the token.

options (RouteOptions, optional)

Additional options for customizing the route. This is defined by theRouteOptions interface (detailed below, seeRoute Options).

Route Options

TheRouteOptions interface allows for further customization of the route request. Below are the parameters for theRouteOptions interface along with their descriptions:

integrator (string, optional)

The identifier of the integrator, usually the dApp or company name. Ideally, this should be specified when configuring the SDK, but it can also be modified during a request. SeeConfigure SDK .

fee (number, optional)

The integrator fee percentage (e.g., 0.03 represents a 3% fee). This requires the integrator to be verified.

As an integrator, you can monetize your LI.FI SDK integration and collect fees. SeeMonetization / Take Fees for more information.

maxPriceImpact (number, optional)

Hides routes with a price impact greater than or equal to this value. (e.g., 0.3 represents 30%)

order (string, optional)

Default :CHEAPEST

The order preference for routes:

- CHEAPEST
- This sorting option prioritises routes with the highest estimated return amount. Users who value capital efficiency at the expense of speed and route complexity should choose the cheapest routes.
- FASTEST
- This sorting option prioritizes routes with the shortest estimated execution time. Users who value speed and want their transactions to be completed as quickly as possible should choose the fastest routes.

slippage (number, optional)

Default :0.005

The slippage tolerance, expressed as a decimal proportion (e.g., 0.005 represents 0.5%).

referrer (string, optional)

The wallet address of the referrer, for tracking purposes.

allowSwitchChain (boolean, optional)

Default :false

Specifies whether to return routes that require chain switches (2-step routes).

allowDestinationCall (boolean, optional)

Default :true

Specifies whether destination calls are enabled.

bridges (AllowDenyPrefer, optional)

AnAllowDenyPrefer object to specify preferences for bridges.

exchanges (AllowDenyPrefer, optional)

AnAllowDenyPrefer object to specify preferences for exchanges.

timing (Timing, optional)

ATiming object to specify preferences forTiming Strategies .

Allow/Deny/Prefer

TheAllowDenyPrefer interface is used to specify preferences for bridges or exchanges. Using theallow option, you can allow tools, and only those tools will be used to find the best routes. Tools specified indeny will be blocklisted.

You can find all available keys inList: Chains, Bridges, DEX Aggregators, Solvers or get the available option from the API. SeeChains and Tools .

Below are the parameters for theAllowDenyPrefer interface:

allow (string[], optional)

A list of allowed bridges or exchanges (default: all).

deny (string[], optional)

A list of denied bridges or exchanges (default: none).

prefer (string[], optional)

A list of preferred bridges or exchanges (e.g.,['1inch'] to prefer 1inch if available).

Timing

TheTiming interface allows you to specify preferences for the timing of route execution. This can help optimize the performance of your requests based on timing strategies.

Parameters for theTiming interface:

- swapStepTimingStrategies
- (TimingStrategy[], optional)
- An array of timing strategies specifically for each swap step in the route. This allows you to define custom strategies for timing control during the execution of individual swap steps.
- routeTimingStrategies
- (TimingStrategy[], optional)
- An array of timing strategies that apply to the entire route. This enables you to set preferences for how routes are timed overall, potentially improving execution efficiency and reliability.

Timing Strategy

This can help optimize the timing of requests based on specific conditions.

Parameters:

- strategy
- (string)
- The strategy type, which must be set to'minWaitTime'
- . This indicates that the timing strategy being applied is based on a minimum wait time.
- minWaitTimeMs
- (number)
- The minimum wait time in milliseconds before any results are returned. This value ensures that the request waits for a specified duration to allow for more accurate results.
- startingExpectedResults
- (number)
- The initial number of expected results that should be returned after the minimum wait time has elapsed. This helps in managing user expectations regarding the outcomes of the request.
- reduceEveryMs
- (number)
- The interval in milliseconds at which the expected results are reduced as the wait time progresses. This parameter allows for dynamic adjustments to the expected results based on the elapsed time.

You can implementcustom timing strategies to improve the user experience and optimize the performance of your application by controlling the timing of route execution.

Request a Quote

When you request a quote, our smart routing API provides the best available option. The quote includes all necessary information and transaction data required to initiate a swap or bridging transfer.

Here is a simple example of how to request a quote to bridge and swap 10 USDC on Arbitrum to the maximum amount of DAI on Optimism.

```

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

constquoteRequest:QuoteRequest={ fromChain:42161,// Arbitrum toChain:10,// Optimism fromToken:'0xaf88d065e77c8cC2239327C5EDb3A432268e5831',// USDC on Arbitrum toToken:'0xDA10009cBd5D07dd0CeCc66161FC93D7c9000da1',// DAI on Optimism fromAmount:'10000000',// 10 USDC // The address from which the tokens are being transferred. fromAddress:'0x552008c0f6870c2f77e5cC1d2eb9bdff03e30Ea0', };

constquote=awaitgetQuote(quoteRequest);
```

```

Quote request parameters

ThegetQuotes function expects aQuoteRequest object.RoutesRequest andQuoteRequest have some similarities and slight differences, and below, you can find a description of theQuoteRequest interface's parameters.

fromChain (number, required)

The ID of the source chain (e.g., Ethereum mainnet is 1).

fromToken (string, required)

The contract address of the token on the source chain. Ensure this address corresponds to the specifiedfromChain .

fromAmount (string, required)

The amount to be transferred from the source chain, specified in the smallest unit of the token (e.g., wei for ETH).

fromAddress (string, required)

The address from which the tokens are being transferred.

toChain (number, required)

The ID of the destination chain (e.g., Optimism is 10).

toToken (string, required)

The contract address of the token on the destination chain. Ensure this address corresponds to the specifiedtoChain .

toAddress (string, optional)

The address to which the tokens will be sent on the destination chain once the transaction is completed.

fromAmountForGas (string, optional)

Part of theLI.Fuel . Allows receiving a part of the bridged tokens as gas on the destination chain. Specified in the smallest unit of the token.

Other Quote parameters

In addition to the parameters mentioned above, all parameters listed in theRoute Options section are also available when usinggetQuote , except forallowSwitchChain , which is used exclusively to control chain switching in route requests.

Also, parameters to specify options for allowing, denying, or preferring certain bridges and exchanges have slightly different names:

- allowBridges
- (string[], optional)
- denyBridges
- (string[], optional)
- preferBridges
- (string[], optional)
- allowExchanges
- (string[], optional)
- denyExchanges
- (string[], optional)
- preferExchanges
- (string[], optional)

Additionally, you can specifytiming strategies for the swap steps using theswapStepTimingStrategies parameter:

- swapStepTimingStrategies
- (string[], optional)
- Specifies the timing strategy for swap steps. This parameter allows you to define how long the request should wait for results and manage expected outcomes. The format is:
- ```
- Copy
- minWaitTime-{minWaitTimeMs}-{startingExpectedResults}-{reduceEveryMs}
- ```

Request contract call Quote

Besides requesting general quotes, the LI.FI SDK also provides functionality to request quotes for destination contract calls.

Read moreTX-Batching aka "Zaps" .

Here is a simple example of how to request a quote to bridge and purchase an NFT on the OpenSea marketplace costing 0.0000085 ETH on the Base chain using ETH from Optimism. The call data for this example was obtained using the OpenSea Seaport SDK.

```

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

constcontractCallsQuoteRequest={ fromAddress:'0x552008c0f6870c2f77e5cC1d2eb9bdff03e30Ea0', fromChain:10, fromToken:'0x0000000000000000000000000000000000000000', toAmount:'8500000000000', toChain:8453, toToken:'0x0000000000000000000000000000000000000000', contractCalls:[{ fromAmount:'8500000000000', fromTokenAddress:'0x0000000000000000000000000000000000000000', toContractAddress:'0x0000000000000068F116a894984e2DB1123eB395', toContractCallData: '0xe7acab2400000000000000000000000000000000000000000000000000000000000000800000000000000000000000000000000000000000000000000000000000000006e0000000000000000000000000 toContractGasLimit:'210000', },], };

constcontractCallQuote=awaitgetContractCallsQuote(contractCallsQuoteRequest);

```

Contract call Quote request parameters

ThegetContractCallsQuote function expects aContractCallsQuoteRequest object, which includes all the information needed to request a quote for a destination contract call.

Contract call quote request can be treated as an extension to the quote request and in addition to the parameters mentioned below, all parameters listed in theOther Quote parameters section (such asintegrator ,fee ,slippage , etc.) are also available when usinggetContractCallsQuote .

fromChain (number, required)

The ID of the source chain (e.g., Ethereum mainnet is 1).

fromToken (string, required)

The contract address of the token on the source chain. Ensure this address corresponds to the specifiedfromChain .

fromAddress (string, required)

The address from which the tokens are being transferred.

toChain (number, required)
```

The ID of the destination chain (e.g., Optimism is 10).

toToken (string, required)

The token required to perform the contract interaction (can be something to stake, donate, or to be used as payment). Ensure this address corresponds to the specifiedtoChain .

toAmount (string, required)

The amount of tokens required by the contract interaction. The LI.FI API will try and generate a quote that guarantees at least that amount on the destination chain.

toFallbackAddress (string, optional)

The address to send the bridged tokens to if the call fails. Defaults to the sending address if not specified.

contractCalls (ContractCall[], required)

Array of contract call objects.

- fromAmount
- (string, required): The amount of tokens to be sent to the contract. This amount is independent of any previously bridged or deposited tokens.
- fromTokenAddress
- (string, required): The address of the token to be sent to the contract. For example, an ETH staking transaction would require ETH.
- toContractAddress
- (string, required): The address of the contract to interact with on the destination chain.
- toContractCallData
- (string, required): The call data to be sent to the contract for the interaction on the destination chain.
- toContractGasLimit
- (string, required): The estimated gas required for the contract call. Incorrect values may cause the interaction to fail.
- toApprovalAddress
- (string, optional): The address to approve the token transfer if it is different from the contract address.
- toTokenAddress
- (string, optional): The address of the token that will be output by the contract, if applicable (e.g., staking ETH produces stETH).

contractOutputsToken (string, optional)

The token address that will be output by the contract, if applicable. Omit if no token should be returned.

Difference betweenRoute andQuote

Even thoughRoute andQuote terms lie in the same field of providing you with the best option to make a swap or bridging transfer, there are some differences you need to be aware of.

ARoute in LI.FI represents a detailed transfer plan that may includemultiple steps . Each step corresponds to an individual transaction, such as swapping tokens or bridging funds between chains. These steps must be executed in a specific sequence, as each one depends on the output of the previous step. ARoute provides a detailed pathway for complex transfers involving multiple actions.

In contrast, aQuote is asingle-step transaction. It contains all the necessary information to perform a transfer in one go, without requiring any additional steps.Quotes are used for simpler transactions where a single action, such as a token swap or a cross-chain transfer, is sufficient. Thus, whileRoutes can involve multiple steps to complete a transfer, aQuote always represents just one step.

Read moreLI.FI Terminology .