Checking the Status of a Transaction

Checking if a cross-chain transfer is completed. The full API documentation for our/status endpoint can be foundhere . If you already looked at ourTransferring Tokens Guide , you know how to request and submit a transaction.

To fetch the execution status of a transfer, the /status endpoint can be queried.

ThetxHash needs to be set and thefromChain ,toChain , andbridge parameters are optional. API will help to calculate other missing parameters and tell whether the transaction is a cross-chain transfer or a simple swap

While non of the parametersfromChain ,toChain andbridge are required, passing thefromChain parameter will speed up the request and is therefore encouraged. If you are willing to provide all parameters for swaps, please note that thefromChain andtoChain have to be set to the same value. Thebridge parameter can be omitted.

```

Copy constgetStatus=async(txHash)=>{ constresult=awaitaxios.get('https://li.quest/v1/status',{ params:{ txHash, } }); returnresult.data; }

```

The result will look similar to the following:

```

Copy { "sending":{ "chainId":1, "txHash":"0x807e84fcea166e316742f811a47c464c02cc1f2cd176f514308244831cc67182", "txLink":"https://etherscan.io/tx/0x807e84fcea166e316742f811a47c464c02cc1f2cd176f514308244831cc67182", "amount":"2000000000000000000", "token":{ ... },// information about the sent token "gasPrice":"204953572552", "gasUsed":"124062" }, "receiving":{ "chainId":42161, "txHash":"0x233114043b0bc383d7d87ab3d8451b24ce17474f26e4c08a9d08ff18f18f639f", "txLink":"https://arbiscan.io/tx/0x233114043b0bc383d7d87ab3d8451b24ce17474f26e4c08a9d08ff18f18f639f", "amount":"2004304477512464378", "token":{ ... },// information about the received token "gasPrice":"1140752846", "gasUsed":"0"

}, "tool":"hop", "status":"DONE", "substatus":"COMPLETE" }

```

The different statuses and what they mean

The response from our /status endpoint contains the propertystatus , indicating the current state of the transaction. The possible values areNOT_FOUND ,INVALID ,PENDING ,DONE , orFAILED .

- NOT_FOUND
- : This means that the transaction doesn't exist. This can happen if the passed hash refers to a transaction that hasn't been mined yet.
- INVALID
- The provided transaction hash does not seem to be tied to the tool you requested the status from. This can happen if the status monitoring is requested for 'connext', but the transaction was processed through 'hop'.
- PENDING
- : The initial transaction was mined, but the bridging process is still in progress. You can get this status also if part of the third-party services we rely on is not reachable or the RPC communication isn't working properly. Refer to the sub-status information to know more.
- DONE
- : The final transaction was received on the destination chain and the operation has concluded. The sub-status will indicate whether there has been a refund or if the tokens received were only partially delivered.
- FAILED
- : The bridging transaction was marked as failed. Look into the transactions to learn what exactly went wrong.
-

The different sub-statuses and what they mean

When the status is eitherDONE ,FAILED orPENDING , we also return a sub-status to explain the situation better.

Sub-statuses ofPENDING :

- WAIT_SOURCE_CONFIRMATIONS
- : The bridge is waiting for additional confirmations.
- WAIT_DESTINATION_TRANSACTION
- : The off-chain logic is in progress, waiting for the destination transaction to be mined. Check back later.

- BRIDGE_NOT_AVAILABLE
- : The bridge API / subgraph is temporarily unavailable, check back later.
- CHAIN_NOT_AVAILABLE
- : The RPC for the source/destination chain is temporarily unavailable
- REFUND_IN_PROGRESS
- : The refund has been requested and it's being processed (not all bridges will go through this state!)
- UNKNOWN_ERROR
- : We cannot determine the status of the transfer.
-

Sub-statuses ofDONE :

- COMPLETED
- : The transfer was successful.
- PARTIAL
- : The transfer was partially successful. This can happen for specific bridges likeacross
- ,hop
- oramarok
- which may provide alternative tokens in case of low liquidity.
- REFUNDED
- : The transfer was not successful and the sent token has been refunded
-

Sub-status ofFAILED :

- NOT_PROCESSABLE_REFUND_NEEDED
- : The transfer cannot be completed, a refund is required.
- OUT_OF_GAS
- : The transaction ran out of gas during execution.
- SLIPPAGE_EXCEEDED
- : The return amount is not enough. This can occur when the slippage limit is exceeded.
- INSUFFICIENT_ALLOWANCE
- : Transfer amount exceeds allowance (insufficient allowance).
- INSUFFICIENT_BALANCE
- : Transfer amount exceeds balance.
- UNKNOWN_ERROR
- : An unknown error occurred or an invalid amount was sent. This can include errors like InvalidAmount, ORACLE_NOT_FEASIBLE, LayerZero: not enough native for fees, and more.
- EXPIRED
- : The transaction expired before it could be processed.
-

Handling unexpected receiving token

- REFUNDED
- : The transfer was not successful and the sent token has been refunded
-

When usingamarok orhop it can happen thatreceiving.token is not the token requested in the original quote:

- amarok
- mints custom nextToken when bridging and swap them automatically to the token representation the user requested. In rare cases, it can happen that while the transfer was executed, the swap liquidity to exchange that token was used up. In this case, the user receives the nextToken instead. You can go to thiswebpage
- to exchange that token later.
- hop
- mints custom hToken when bridging and swap them automatically to the token representation the user requested. In rare cases, it can happen that while the transfer was executed, the swap liquidity to exchange that token was used up. In this case, the user receives the nextToken instead. You can go to thiswebpage
- to exchange that token later.
-

Querying the status endpoint

If your dApp wants to wait for the completion of the cross-chain transfer, it can query the /status endpoint until a final state is reached:

```

Copy constgetStatus=async(txHash)=>{ constresult=awaitaxios.get('https://li.quest/v1/status',{ params:{ txHash, } });
```

returnresult.data; }

letresult; do{ result=awaitgetStatus(tx.hash); }while(result.status!=='DONE'&&result.status!=='FAILED')

console.log(Transfer completed with state{result.status});

``` Last updated5 days ago On this page *The different statuses and what they mean * The different sub-statuses and what they mean * Querying the status endpoint

Was this helpful? Export as PDF
```