Estimating Fees

There are two types of fees paid to offchain agents for each applicablexcall .

- Router Fee
- : 0.05% of the transferred asset will be levied by routers on destination for their service as fast liquidity providers.
-
    - The 'fast path' is possible when routers have liquidity in the destination asset and are able to provide those assets to the user. This allows users to receive their desired destination assets almostimmediately
-
    - .
-
    - Routers will take on the bridge delay and wait for the optimistic period to pass. Once complete, the bridge will 'reconcile' by minting the local destination assets to the routers, making them whole again.
-
    - If anxcall
-
    - goes through the 'slow path' (authenticated), then users do not pay the router fee.
-
    - Note that routers always provideand
-
    - receive minted assets of the local destination flavor - they never have to rebalance funds!
- *
- Relayer Fee
- : A fee charged by relayers on top of normal gas costs in exchange for providing a meta-transaction service.
-
    - Relayers execute transactions on the destination chain on behalf of users.
-
    - Users offer a fee bounty to incentivize relayers to execute their destination calls.
-
    - Relayer fees are paid in the origin native asset or the transacting asset and need to be estimated whenxcall
-
    - is initiated. Some relayers provide endpoints that can help with estimation.
- *
-

Router fees are fixed and hardcoded into the Connext protocol. Relayer fees, on the other hand, can vary between chains and the service provider.

Estimating Relayer Fees

For now, we need to rely on offchain tools to estimate relayer fees. The Connext SDK abstracts away some of this complexity.

TheSdkBase class includes anestimateRelayerFee method that estimates total gas fees including a bump to account for Gelato relayer fees.

The relayer fee can be paid in either the native asset or the transacting asset (the asset being bridged in thexcall).

Pay in native asset

The resulting estimate will be converted to the native origin asset.

```

Copy const{sdkBase}=awaitcreate(nxtpConfig);

constparams={ originDomain:"", destinationDomain:"", }

constrelayerFee=awaitsdkBase.estimateRelayerFee(params);

```

The estimate should be used as therelayerFee param for anxcall using the SDK.

```

Copy constxcallTxReq=awaitsdkBase.xcall( ..., relayerFee: relayerFee );

```

Or passed in as thevalue for anxcall in a smart contract.

```

Copy contractSource{ ... functioncrossChainCall() { ... connext.xcall{value:relayerFee}(...); } }

```

Pay in transacting asset

The resulting estimate will be the relayer fee in USD.

```

Copy const{sdkBase}=awaitcreate(nxtpConfig);

constparams={ originDomain:"", destinationDomain:"", priceIn:"usd"// use this if you want the estimate in USD }

constrelayerFeeInTransactingAsset=awaitsdkBase.estimateRelayerFee(params);

```

The estimate in USD should be converted to the value of the transacting asset (e.g. by using a price feed) and supplied as therelayerFeeInTransactingAsset param for anxcall using the SDK.

```

Copy constxcallTxReq=awaitsdkBase.xcall( ..., relayerFeeInTransactingAsset: relayerFeeInTransactingAsset );

```

Or passed in as the_relayerFee for anxcall in a smart contract.

```

Copy contractSource{ ... functioncrossChainCall() { ... connext.xcall( ..., relayerFeeInTransactingAsset ); } }

```

Bumping Relayer Fees

Since gas conditions are impossible to predict, transactions can potentially stay pending on destination if fees aren't high enough. Connext allows the user (or anyone if they are feeling charitable) to increase the original fee until sufficient for relayers.

Anyone can call the Connext contract functionbumpTransfer to increase the original relayer fee for anxcall .

Bump in native asset

To bump using SDK:

```

Copy constbumpTxReq=awaitsdkBase.bumpTransfer( domainId: originDomain, transferId: transferId, asset:, relayerFee: );

```

To bump from a contract call:

```

Copy functionbumpTransfer(bytes32_transferId)externalpayable;

```

Bump in transacting asset

To bump using SDK:

```

Copy constbumpTxReq=awaitsdkBase.bumpTransfer( domainId: originDomain, transferId: transferId, asset:, relayerFee: );
```

```

To bump from a contract call:

```

Copy functionbumpTransfer(bytes32_transferId,address_relayerFeeAsset,uint256_relayerFee)externalpayable;

```

To find thetransferId , see Tracking xCalls.

Previous SDK Next Tracking xCalls Last updated9 months ago On this page *Estimating Relayer Fees * Pay in native asset * Pay in transacting asset * Bumping Relayer Fees * Bump in native asset * Bump in transacting asset

Edit on GitHub
```