CCIP Best Practices

Before you deploy your cross-chain dApps to mainnet, make sure that your dApps follow the best practices in this document. You are responsible for thoroughly reviewing your code and applying best practices to ensure that your cross-chain dApps are secure and reliable. If you have a unique use case for CCIP that might involve additional cross-chain risk,contact the Chainlink Labs Team before deploying your application to mainnet.

Interfaces and Applications

Chainlink CCIP is a messaging protocol. Third parties may build user interfaces or other applications on top of CCIP. Neither Chainlink Labs nor the Chainlink Foundation owns, controls, endorses, or assumes any responsibility for any such interfaces or applications. You are solely responsible for your use of such interfaces or applications. Please visit the Chainlink FoundationTerms of Service for more information.

Verify destination chain

Before calling the router'sccipSendfunction , ensure that your code allows users to send CCIP messages to trusted destination chains.

Verify source chain

When implementing theccipReceivemethod in a contract residing on the destination chain, ensure to verify the source chain of the incoming CCIP message. This verification ensures that CCIP messages can only be received from trusted source chains.

Verify sender

When implementing theccipReceivemethod in a contract residing on the destination chain, it's important to validate the sender of the incoming CCIP message. This check ensures that CCIP messages are received only from trusted sender addresses.

Note: Depending on your use case, this verification might not always be necessary.

Verify router addresses

When you implement theccipReceivemethod in the contract residing on the destination chain, validate that themsg.senderis the correct router address. This verification ensures that only the router contract can call theccipReceivefunction on the receiver contract and is for developers that want to restrict which accounts are allowed to callccipReceive.

SettinggasLimit

ThegasLimitspecifies the maximum amount of gas CCIP can consume to executeccipReceive()on the contract located on the destination blockchain. It is the main factor in determining the fee to send a message. Unspent gas is not refunded.

To transfer tokens directly to an EOA as areceiveron the destination blockchain, thegasLimitshould be set to0since there is noccipReceive()implementation to call.

To estimate the accurate gas limit for your destination contract, consider the following options:

- Leveraging Ethereum client RPC by applyingeth_estimateGasonreceiver.ccipReceive(). You can find more information on theEthereum API Documentation andAlchemy documentation .
- ConductingFoundry gas tests .
- UsingHardhat plugin for gas tests .
- Using a blockchain explorer to look up the gas consumption of a particular internal transaction.

UsingextraArgs

The purpose ofextraArgsis to allow compatibility with future CCIP upgrades. To get this benefit, make sure thatextraArgsis mutable in production deployments. This allows you to build it offchain and pass it in a call to a function or store it in a variable that you can update on-demand.

IfextraArgsare left empty, a default of200000gasLimitwill be set.

Decoupling CCIP Message Reception and Business Logic

As a best practice, separate the reception of CCIP messages from the core business logic of the contract. Implement 'escape hatches' or fallback mechanisms to gracefully manage situations where the business logic encounters issues. To explore this concept further, refer to theDefensive Example guide.

Evaluate the security and reliability of the networks that you use

Although CCIP has been thoroughly reviewed and audited, inherent risks might still exist based on your use case, the blockchain networks where you deploy your contracts, and the network conditions on those blockchains.

Review and audit your code

Before securing value with contracts that implement CCIP interfaces and routers, ensure that your code is secure and reliable. If you have a unique use case for CCIP that might involve additional cross-chain risk,contact the Chainlink Labs Team before deploying your application to mainnet.

Soak test your dApps

Be aware of theService Limits and Rate Limits for Supported Networks . Before you provide access to end users or secure value, soak test your cross-chain dApps. Ensure that your dApps can operate within these limits and operate correctly during usage spikes or unfavorable network conditions.

Monitor your dApps

When you build applications that depend on CCIP, include monitoring and safeguards to protect against the negative impact of extreme market events, possible malicious activity on your dApp, potential delays, and outages.

Create your own monitoring alerts based on deviations from normal activity. This will notify you when potential issues occur so you can respond to them.