CCIP Architecture

Prerequisites

Read the CCIPIntroduction andConcepts to understand all the concepts discussed on this page.

High-level architecture

Below is a diagram displaying the basic architecture of CCIP. Routers are smart contracts that provide a simple and consistent interface for users. Users can interact with routers to:

- Call smart contract functions on a different blockchain.
- Transfer tokens to a smart contract orExternally Owned Account (EOA) on a different blockchain.
- Send arbitrary messages and tokens within the same transaction. Use this functionality to transfer tokens and instructions on what to do with those tokens to a smart contract on a different blockchain.

Transferring tokens

This section uses the term "transferring tokens" even though the tokens are not technically transferred. Instead, they are locked or burned on the source chain and then unlocked or minted on the destination chain. Read theToken Pools section to understand the various mechanisms that are used to transfer value across chains.

CCIP terminology

CCIP enables asenderon asource blockchainto send amessageto areceiveron adestination blockchain.

TermDescriptionSenderA smart contract or an EOA.Source BlockchainThe blockchain the sender interacts with CCIP from.MessageArbitrary data and/or tokens.ReceiverA smart contract or an EOA.Note: An EOA cannot receive arbitrary data. It can only receive tokens.Destination BlockchainThe blockchain the receiver resides on.

Detailed architecture

The figure below outlines the different components involved in a cross-chain transaction:

- Cross-Chain dApps are user-specific. A smart contract or anEOA (Externally Owned Account) interacts with the CCIP Router to send arbitrary data and/or transfer tokens cross-chain.
- The contracts in dark blue are the CCIP interface (Router). To use CCIP, users need only to understand how to interact with the router; they don't need to understand the whole CCIP architecture.Note: The CCIP interface is static and remains consistent over time to provide reliability and stability to the users.
- The contracts in light blue are internal to the CCIP protocol and subject to change.

Onchain components

Router

The Router is the primary contract CCIP users interface with. This contract is responsible for initiating cross-chain interactions. One router contract exists per chain. When transferring tokens, callers have to approve tokens for the router contract. The router contract routes the instruction to the destination-specificOnRamp .

When a message is received on the destination chain, the router is the contract that “delivers” tokens to the user's account or the message to the receiver's smart contract.

Commit Store

TheCommitting DON interacts with theCommitStorecontract on the destination blockchain to store the Merkle root of the finalized messages on the source blockchain. This Merkle root must be blessed by theRisk Management Network before theExecuting DON can execute them on the destination blockchain. TheCommitStoreensures the message is blessed by theRisk Management Network and only oneCommitStoreexists perlane .

OnRamp

One OnRamp contract perlane exists. This contract performs the following tasks:

- Checks destination-blockchain-specific validity such as validating account address syntax
- Verifies the message size limit and gas limits
- Keeps track of sequence numbers to preserve the sequence of messages for the receiver

- Managesbilling
- Interacts with theTokenPool if the message includes a token transfer.
- Emits an event monitored by thecommitting DON

OffRamp

One OffRamp contract perlane exists. This contract performs the following tasks:

- Ensures the message is authentic by verifying the proof provided by the Executing DON against a committed and blessed Merkle root
- Makes sure transactions are executed only once
- After validation, the OffRamp contract transmits any received message to theRouter contract. If the CCIP transaction includes token transfers, the OffRamp contract calls theTokenPool to transfer the correct assets to the receiver.

Token pools

Each token has its own token pool, an abstraction layer over ERC-20 tokens that facilitates OnRamp and OffRamp token-related operations. Token pools are configurable tolockorburnat the source blockchain andunlockormintat the destination blockchain. The mechanism for handling tokens depends on the characteristics of the token in question. Here are a few examples:

- Blockchain-native gas tokens, such as ETH, MATIC, and AVAX, can only be minted on their native chains. These tokens cannot be burned on the source and minted at the destination to transfer these tokens between chains. Instead, the linked token pool uses a "Lock and Mint" approach that locks the token at its source and then mints a wrapped or synthetic asset on the destination blockchain. This synthetic asset represents the locked asset and is essential for redeeming the locked asset.
- A token like LINK is minted on a single chain (Ethereum mainnet) with a fixed total supply. CCIP cannot natively mint it on another chain. In this case, the "Lock and Mint" approach is required.
- Some tokens can be minted on multiple chains. Examples of such tokens include stablecoins like USDC, TUSD, USDT, and FRAX. The linked token pools use a "Burn and Mint" method to burn the token at its source and then mint it natively on the destination blockchain. Wrapped assets such as WBTC or WETH are other examples that use the "Burn and Mint" approach.
- A token with a Proof Of Reserve (PoR) feed on a specific chain poses a challenge for the "Burn and Mint" method when applied to other chains because it conflicts with the PoR feed. For these tokens, "Lock and Mint" is the preferred approach.

Token pools provide rate limiting, which is a security feature enabling token issuers to set a maximum rate at which their token can be transferred.

Onboarding New Tokens

Supported tokens for each lane are listed on theCCIP supported network pages . Expanding support for additional tokens is an ongoing process to help ensure the highest level of cross-chain security.Token pool rate limits are configured per-token for each lane. These rate limits are set together with token issuers where applicable. Rate limits are also selected based on various factors, such as the unique risk characteristics of the token and the intended use cases. The onboarding process for adding new tokens also helps ensure that dApps and users interact with the correct token version, reducing risks within the cross-chain ecosystem.

Risk Management Network contract

The Risk Management contract maintains the list of Risk Management node addresses that are allowed to bless or curse. The contract also holds the quorum logic for blessing a committed Merkle Root and cursing CCIP on a destination blockchain. Read theRisk Management Network Concepts section to learn more.

Offchain components

Committing DON

The Committing DON has several jobs where each job monitors cross-chain transactions between a given source blockchain and destination blockchain:

- Each job monitors events from a givenOnRamp contract on the source blockchain.
- The job waits forfinality , which depends on the source blockchain.
- The job bundles transactions and creates a Merkle root. This Merkle root is signed by a quorum of oracles nodes part of the Committing DON.
- Finally, the job writes the Merkle root to theCommitStore contract on the given destination blockchain.

Executing DON

Like the Committing DON , the Executing DON has several jobs where each executes cross-chain transactions between a source blockchain and a destination blockchain:

- Each job monitors events from a given OnRamp contract on the source blockchain.
- The job checks whether the transaction is part of the relayed Merkle root in the CommitStore contract .
- The job waits for the Risk Management Network to bless the message.
- Finally, the job creates a valid Merkle proof, which is verified by the OffRamp contract against the Merkle root in the CommitStore contract . After these check pass, the job calls the OffRamp contract to complete the CCIP transactions on the destination blockchain.

Separating commitment and execution permits the Risk Management Network to have enough time to check the commitment of messages before executing them. The delay between commitment and execution also permits additional checks such as abnormal reorg depth, potential simulation, and slashing.

Saving a commitment is compact and has a fixed gas cost, whereas executing user callbacks can be highly gas intensive. Separating commitment and execution permits execution by end users in various cases, such as retrying failed executions.

Risk Management Network

The Risk Management Network is a set of independent nodes that monitor the Merkle roots committed by the Committing DON into the Commit Store .

Each node compares the committed Merkle roots with the transactions received by the OnRamp contract . After the verification succeeds, it calls the Risk Management contract to"bless"the committed Merkle root. When there are enough blessing votes, the root becomes available for execution. In case of anomalies, each Risk Management node calls the Risk Management contract to"curse"the system. If the cursed quorum is reached, the Risk Management contract is paused to prevent any CCIP transaction from being executed.

Read the Risk Management Network Concepts section to learn more.

CCIP rate limits

Chainlink CCIP token transfers benefit from rate limits for additional security. A rate limit has a maximum capacity and a refill rate, which is the speed at which the maximum capacity is restored after a token transfer has consumed some or all of the available capacity.

You can find the complete list of lanes and their rate limits on the CCIP Supported Networks page.

The rate limits are enforced at both the source and destination blockchains for maximum security. If these rate limits are reached, descriptive errors with detailed information are generated and returned to the sender. This enables CCIP users to gracefully handle these errors within their dApps to preserve the end-user experience. A comprehensive list of errors and their descriptions is available on the errors API reference page.

Token pool rate limit

For each supported token on each individual lane , the token pool rate limit manages the total number of tokens that can be transferred within a given time. This limit is independent of the USD value of the token.

For example, the maximum capacity of the suUSD token pool on the Ethereum mainnet → Base mainnet lane is 200,000 suUSD. The refill rate is 2 suUSD per second. If 200,000 suUSD are transferred on that lane, the entire capacity is consumed. After that, if a user wants to send 20 suUSD, they must wait at least 10 seconds as the capacity refills. The maximum throughput for this token on the lane is 1200 suUSD every 10 minutes.

Aggregate rate limit

Each lane also has an aggregate rate limit that limits the overall USD value that can be transferred on that lane across all supported tokens. To improve security, the aggregate rate limit for any given lane is always lower than the sum of all individual token pool rate limits for that lane.

Consider an example where a lane has a maximum capacity of 100,000 USD, a refill rate of 167 USD per second, and several token transfers with a total value of 60,000 USD have been executed. In that example, the remaining available capacity is 40,000 USD. If a user intends to transfer tokens equating to 50,000 USD, they must wait at least 60 seconds for capacity to refill the additional 10,000 USD that is required. The maximum throughput in USD value on the lane is 100,000 USD every 10 minutes.

CCIP execution latency

Chainlink CCIP has been purposely designed to take a security-first approach to minimize the risk of block reorgs on the source blockchain. The end-to-end transaction time of a CCIP cross-chain transaction largely depends on the time it takes for the transaction on the source chain to reachfinality . The time to reach finality varies by blockchain. For example, on Ethereum, it takes about15 minutes for a block to be finalized. When cross-chain transactions are initiated from a chain with faster finality, such as Avalanche, which has atime-to-finality of around one second , the end-to-end transaction time is faster.

If the fee paid on the source chain is within an acceptable range of the execution cost on the destination chain, the message will be transmitted as soon as possible after it is blessed by the Risk Management Network. If the cost of execution increases between request and execution time, CCIP incrementally increases the gas price to attempt to reach eventual execution, but this might introduce additional latency.

Execution delays in excess of one hour should be rare as a result ofSmart Execution. The Smart Execution time window parameter represents the waiting time before manual execution is enabled. If the DON fails to execute within the duration of Smart Execution time window due to extreme network conditions, you can manually execute the message through theCCIP Explorer . Read themanual execution conceptual guide to learn more.