Subscription Method

You are viewing the VRF v2 guide - Subscription method

To learn how to request random numbers without a subscription, see thedirect funding method guide.

Security Considerations

Be sure to review your contracts with thesecurity considerations in mind.

This section explains how to generate random numbers using the subscription method.

Subscriptions

VRF v2 requests receive funding from subscription accounts. TheSubscription Manager lets you create an account and pre-pay for VRF v2, so you don't provide funding each time your application requests randomness. This reduces the total gas cost to use VRF v2. It also provides a simple way to fund your use of Chainlink products from a single location, so you don't have to manage multiple wallets across several different systems and applications.

Subscription Manager

Read theSubscription Manager UI page to learn how to use all the features of the VRF v2 user interface. To learn how to troubleshoot your VRF requests, read thepending andfailed requests sections.

Go tovrf.chain.link to open the Subscription Manager.

Subscriptions have the following core concepts:

- Subscription id:64-bit unsigned integer representing the unique identifier of the subscription.
- Subscription accounts:An account that holds LINK tokens and makes them available to fund requests to Chainlink VRF v2 coordinators.
- Subscription owner:The wallet address that creates and manages a subscription account. Any account can add LINK to the subscription balance, but only the owner can add approved consuming contracts or withdraw funds.
- Consumers:Consuming contracts that are approved to use funding from your subscription account.
- Subscription balance:The amount of LINK maintained on your subscription account. Requests from consuming contracts will continue to be funded until the balance runs out, so be sure to maintain sufficient funds in your subscription balance to pay for the requests and keep your applications running.

For Chainlink VRF v2 to fulfill your requests, you must maintain a sufficient amount of LINK in your subscription balance. Gas cost calculation includes the following variables:

- Gas price:The current gas price, which fluctuates depending on network conditions.
- Callback gas:The amount of gas used for the callback request that returns your requested random values.
- Verification gas:The amount of gas used to verify randomness onchain.

The gas price depends on current network conditions. The callback gas depends on your callback function, and the number of random values in your request. The cost of each request is final only after the transaction is complete, but you define the limits you are willing to spend for the request with the following variables:

- Gas lane:The maximum gas price you are willing to pay for a request in wei. Define this limit by specifying the appropriatekeyHashin your request. The limits of each gas lane are important for handling gas price spikes when Chainlink VRF bumps the gas price to fulfill your request quickly.
- Callback gas limit:Specifies the maximum amount of gas you are willing to spend on the callback request. Define this limit by specifying thecallbackGasLimitvalue in your request.

Request and receive data

Requests to Chainlink VRF v2 follow the request and receive data cycle. This end-to-end diagram shows each step in the lifecycle of a VRF subscription request, and registering a smart contract with a VRF subscription account:

Two types of accounts exist in the Ethereum ecosystem, and both are used in VRF:

- EOA (Externally Owned Account): An externally owned account that has a private key and can control a smart contract. Transactions can only be initiated by EOAs.
- Smart contract: A contract that does not have a private key and executes what it has been designed for as a decentralized application.

The Chainlink VRF v2 solution uses both offchain and onchain components:

- VRF v2 Coordinator (onchain component) : A contract designed to interact with the VRF service. It emits an event when a request for randomness is made, and then verifies the random number and proof of how it was generated by the VRF service.
- VRF service (offchain component): Listens for requests by subscribing to the VRF Coordinator event logs and calculates a random number based on the block hash and nonce. The VRF service then sends a transaction to theVRFCoordinatorincluding the random number and a proof of how it was generated.

Set up your contract and request

Set up your consuming contract:

1. Your contract must inheritVRFConsumerBaseV2 .
2. Your contract must implement thefulfillRandomWordsfunction, which is thecallback VRF function. Here, you add logic to handle the random values after they are returned to your contract.

3. Submit your VRF request by callingrequestRandomWordsof theVRF Coordinator . Include the following parameters in your request:

4. keyHash: Identifier that maps to a job and a private key on the VRF service and that represents a specified gas lane. If your request is urgent, specify a gas lane with a higher gas price limit. The configuration for your network can be foundhere .

5. s_subscriptionId: The subscription ID that the consuming contract is registered to. LINK funds are deducted from this subscription.
6. requestConfirmations: The number of block confirmations the VRF service will wait to respond. The minimum and maximum confirmations for your network can be foundhere .
7. callbackGasLimit: The maximum amount of gas a user is willing to pay for completing the callback VRF function. Note that you cannot put a value larger than maxGasLimit of the VRF Coordinator contract (readcoordinator contract limits for more details).
8. numWords: The number of random numbers to request. The maximum random values that can be requested for your network can be foundhere .

How VRF processes your request

After you submit your request, it is processed using theRequest & Receive Data cycle. The VRF coordinator processes the request and determines the final charge to your subscription using the following steps:

1. The VRF coordinator emits an event.
2. The VRF service picks up the event and waits for the specified number of block confirmations to respond back to the VRF coordinator with the random values and a proof (requestConfirmations).
3. The VRF coordinator verifies the proof onchain, then it calls back the consuming contractfulfillRandomWordsfunction.

Limits

Chainlink VRF v2 has somesubscription limits andcoordinator contract limits .

Subscription limits

Subscriptions are required to maintain a minimum balance, and they can support a limited number of consuming contracts.

Minimum subscription balance

Each subscription must maintain a minimum balance to fund requests from consuming contracts. This minimum balance requirement serves as a buffer against gas volatility by ensuring that all your requests have more than enough funding to go through. If your balance is below the minimum, your requests remain pending for up to 24 hours before they expire. After you add sufficient LINK to a subscription, pending requests automatically process as long as they have not expired.

In the Subscription Manager, the minimum subscription balance is displayed as theMax Cost, and it indicates the amount of LINK you need to add for a pending request to process. After the request is processed, only the amount actually consumed by the request is deducted from your balance. For example, if your minimum balance is 10 LINK, but your subscription balance is 5 LINK, you need to add at least 5 more LINK for your request to process. This does not mean that your request will ultimately cost 10 LINK. If the request ultimately costs 3 LINK after it has processed, then 3 LINK is deducted from your subscription balance.

The minimum subscription balance must be sufficient for each new consuming contract that you add to a subscription. For example, the minimum balance for a subscription that supports 20 consuming contracts needs to cover all the requests for all 20 contracts, while a subscription with one consuming contract only needs to cover that one contract.

For one request, the required size of the minimum balance depends on the gas lane and the size of the request. For example, a consuming contract that requests one random value will require a smaller minimum balance than a consuming contract that requests 50 random values. In general, you can estimate the required minimum LINK balance using the following formula where max verification gas is always 200,000 gwei.

(((Gas lane maximum * (Max verification gas + Callback gas limit)) / (1,000,000,000 Gwei/ETH)) / (ETH/LINK price)) + LINK premium = Minimum LINK Here is the same formula, broken out into steps:

Gas lane maximum * (Max verification gas + Callback gas limit) = Total estimated gas (Gwei) Total estimated gas (Gwei) / 1,000,000,000 Gwei/ETH = Total estimated gas (ETH) Total estimated gas (ETH) / (ETH/LINK price) = Total estimated gas (LINK) Total estimated gas (LINK) + LINK premium = Minimum subscription balance (LINK)

Maximum consuming contracts

Each subscription supports up to 100 consuming contracts. If you need more than 100 consuming contracts, create multiple subscriptions.

Coordinator contract limits

You can see the configuration for each network on theSupported networks page. You can also view the full configuration for each coordinator contract directly in Etherscan. As an example, view theEthereum Mainnet VRF v2 coordinator contract configuration.

- Each coordinator has aMAX_NUM_WORDSparameter that limits the maximum number of random values you can receive in each request.
- Each coordinator has amaxGasLimitparameter, which is the maximum allowedcallbackGasLimitvalue for your requests. You must specify a sufficientcallbackGasLimitto fund the callback request to your consuming contract. This depends on the number of random values you request and how you process them in yourfulfillRandomWords()function. If yourcallbackGasLimitis not sufficient, the callback fails but your subscription is still charged for the work done to generate your requested random values.