Direct Funding Method

You are viewing the VRF v2 guide - Direct funding method.

To learn how to request random numbers with a subscription, see theSubscription Method guide.

Security Considerations

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

This guide explains how to generate random numbers using thedirect fundingmethod. This method doesn't require a subscription and is optimal for one-off requests for randomness. This method also works best for applications where your end-users must pay the fees for VRF because the cost of the request is determined at request time.

VRF direct funding

Unlike thesubscription method , the direct funding method does not require you to create subscriptions and pre-fund them. Instead, you must directly fund consuming contracts with LINK tokens before they request randomness. Because the consuming contract directly pays the LINK for the request, the cost is calculated during the request and not during the callback when the randomness is fulfilled. Learnhow to estimate costs .

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 direct funding request:

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 be initiated only by EOAs.
- Smart contract: A smart 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 Wrapper (onchain component) : A wrapper for the VRF Coordinator that provides an interface for consuming contracts.
- 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 inheritVRFV2WrapperConsumerBase .
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 calling therequestRandomnessfunction in theVRFV2WrapperConsumerBase contract. Include the following parameters in your request:

4. requestConfirmations: The number of block confirmations the VRF service will wait to respond. The minimum and maximum confirmations for your network can be foundhere .

5. callbackGasLimit: The maximum amount of gas to pay for completing the callback VRF function.
6. numWords: The number of random numbers to request. You can find the maximum number of random values per request for your network in theSupported networks page.

How VRF processes your request

After you submit your request, it is processed using theRequest & Receive Data cycle:

1. The consuming contract calls theVRFV2Wrapper calculateRequestPricefunction to estimate the total transaction cost to fulfill randomness. Learnhow to estimate transaction costs.

2. The consuming contract calls theLinkToken transferAndCallfunction to pay the wrapper with the calculated request price. This method sends LINK tokens and executes theVRFV2Wrapper onTokenTransferlogic.
3. The VRFV2Wrapper'sonTokenTransferlogic triggers theVRF Coordinator requestRandomWordsfunction to request randomness.
4. The VRF coordinator emits an event.
5. 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).
6. The VRF coordinator verifies the proof onchain, then it calls back the wrapper contract'sfulfillRandomWordsfunction.
7. Finally, the VRF Wrapper calls back your consuming contract.

Limits

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

- Each wrapper has amaxNumWordsparameter that limits the maximum number of random values you can receive in each request.
- The maximum allowedcallbackGasLimitvalue for your requests is defined in theCoordinator contract supported networks page. Because the VRF v2 Wrapper adds an overhead, yourcallbackGasLimitmust not exceedmaxGasLimit - wrapperGasOverhead. Learn more aboutestimating costs .