

Basic Request Model

[Contracts overview](#)

All source code is open source and available in the [Chainlink Github repository](#).

[ChainlinkClient](#)

[ChainlinkClient](#) is a parent contract that enables smart contracts to consume data from oracles. It's available in the Chainlink smart contract library which can be [installed using the latest package managers](#).

The client constructs and makes a request to a known Chainlink oracle through the `transferAndCall` function, implemented by the LINK token. This request contains encoded information that is required for the cycle to succeed. In the `ChainlinkClient` contract, this call is initiated with a call to `sendChainlinkRequestTo`.

To build your own client contract using `ChainlinkClient`, see [Introduction to Using Any API](#), or view the [ChainlinkClient API Reference](#) for the `ChainlinkClient` contract.

[LINK Token](#)

LINK is an [ERC-677](#) compliant token which implements `transferAndCall`, a function that allows tokens to be transferred whilst also triggering logic in the receiving contract within a single transaction.

Learn more about [ERC-677 and the LINK token](#).

[Oracle Contract](#)

[Oracle contracts](#) are owned by oracle node operators, which run alongside offchain oracle nodes.

[Request](#)

The client contract that initiates this cycle must create a request with the following items:

- The oracle address.
- The job ID, so the oracle knows which tasks to perform.
- The callback function, which the oracle sends the response to.

To learn about how to find oracles to suit your needs, see [Find Existing Jobs](#).

Oracle contracts are responsible for handling onchain requests made through the LINK token, by implementing `onTokenTransfer` as a [LinkTokenReceiver](#). Upon execution of this function, the oracle contract emits an `OracleRequest` event containing information about the request. This event is crucial, as it is monitored by the offchain oracle node which acts upon it.

[Fulfillment](#)

For fulfillment, the oracle contract has a `fulfillOracleRequest` function which is used by the node to fulfill a request once it has the result of the job. This function returns the result to the `ChainlinkClient` using the callback function defined in the original request.

[Offchain oracle node](#)

The offchain oracle node is responsible for listening for events emitted by its corresponding onchain smart contract. Once it detects an `OracleRequest` event, it uses the data emitted to perform a job.

The most common job type for a Node is to make a GET request to an API, retrieve some data from it, parse the response, convert the result into blockchain compatible data, then submit it in a transaction back to the oracle contract, using the `fulfillOracleRequest` function.

For more information on how to become a node operator, learn how to [run a Chainlink node](#).

[Consumer UML](#)

Below is a UML diagram describing the contract structure of `ATestnetConsumer`, a deployed example contract implementing `ChainlinkClient`.