Introduction

Welcome to the Carbon Interface documentation 👋

You're here because you want to make your application more climate-concious. Our goal at Carbon Interface is to make it easy for developers to integrate **scientifically-backed** carbon emissions estimates into their applications. By doing this, we can build software that reduces carbon emissions, slowing climate change.

Carbon Interface offers two APIs:

Estimates API

The Estimates API makes it easy to generate accurate emissions estimates from flights, vehicles, shipping, electricity consumption and fuel combustion.

Carbon Ledger API

The Carbon Ledger API makes it easy to generate accurate emissions estimates from credit card and debit transactions.

> Carbon Ledger API is currently in beta. You are able to register and receive stubbed estimates. Please contact ledger@carboninterface.com to request production access.

Getting Started

Carbon Interface's APIs are based on REST and rely on JSON for sending and receiving payloads.

The API will work with programming languages and frameworks that can make standard HTTP requests.

Postman

You can download and test all of the API endpoints in the Postman file below. Postman makes it easy to test APIs before integrating them into your application.

Download Carbon Interface Postman Collection

Authentication

Authentication is done via the API key which you can find in the developer portal. To make a request, the API key must be included as a bearer token in the authorization header.

Carbon Interface expects the API key to be included in all API requests to the server in a header that looks like the following:

```
Authorization: Bearer API_KEY
```

You must replace `API_KEY` with your API key.

To test your API key:

```shell
curl "https://www.carboninterface.com/api/v1/auth"
  -H "Authorization: Bearer API_KEY"
```

If a valid `API_KEY` was used to make the above request, you will receive the following response:

```json
{
  "message": "auth successful"
}
```

Status Codes

The Carbon Interface API returns standard HTTP success or error status codes. For errors, the API will also include extra information about what went wrong encoded in the response as JSON. The various HTTP status codes that might be returned are listed below.

Status Code	Meaning
200	OK
201	Object Created
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
422	Unprocessable Entity
429	Too Many Requests
500	Internal Server Error
503	Service Unavailable

Errors will be returned in the following JSON format:

```json
{
  "message": "The fuel_source_unit provided does not exist for the fuel_s
}
```

The API will provide a descriptive error message in the `message` value.

Estimates API

The Estimates API makes it easy to perform CO_2e emissions estimates for common CO_2e emitting activities.

Carbon Interface makes it easy to get accurate carbon estimates:

1. Provide a details on the activity that is emitting carbon
2. The Estimates API will use the most accurate estimation methodology to compute the amount of CO_2e emitted
3. A JSON response will be returned for you to use in your app

Estimates are calculated by leading CO_2e emissions estimates methodologies from scientists around the world. View our methodology document for more information on how emissions estimates are calculated.

Electricity

The electricity estimate endpoint will provide carbon emissions in grams, pounds, kilograms and tonnes. To create an electricity estimate, provide the following params:

Parameter	Type	Description
type	string	`electricity`
electricity_unit *optional*	string	Can be either megawatt hours `mwh` or kilowatt hours `kwh`. If a unit is not provided, `mwh` is the default.
electricity_value	decimal	Value of the unit of electricity consumption or generation noted above.
country	string	Set to the country you need emissions data on for electricity consumption. Currently, Carbon Interface supports North American and European countries and sub-regions. Visit our geographic coverage page to see what countries and regions are supported. The country

Parameter	Type	Description
		must be passed through using the country's ISO 3166 country code found here.
state *optional*	string	Proving the state will allow the API to generate a more accurate eletrcitiy emissions estimate. Emissions data from electricity generation from all Canadian provinces and US states is supported. The state param needs to be the two letter ISO state code.

Create an Electricity Estimate Request

shell

```
curl "https://www.carboninterface.com/api/v1/estimates"
  -H "Authorization: Bearer API_KEY"
  -H "Content-Type: application/json"
  -X POST
  -d {
        "type": "electricity",
        "electricity_unit": "mwh",
        "electricity_value": 42,
        "country": "us",
        "state": "fl"
      }
```

Create an Electricity Estimate Response

json

```
{
  "data": {
    "id": "2d968fce-859d-4dc1-9489-987e795f42bb",
    "type": "estimate",
    "attributes": {
      "country": "us",
      "state": "fl",
      "electricity_unit": "mwh",
```

```
        "electricity_value": "42.0",
        "estimated_at": "2020-07-24T02:23:24.441Z",
        "carbon_g": 18051428,
        "carbon_lb": 39796,
        "carbon_kg": 18051,
        "carbon_mt": 18
      }
    }
  }
```


Flight

The flight estimate request will accept passenger flight details and compute the carbon emissions for the trip. To compute the carbon emissions of a round trip ensure the `legs` param includes both the departure flight and the return flight.

Parameter	Type	Description
type	string	`flight`
passengers	integer	Number of passengers you want the emissions estimate to be calculated for.
legs	array	array of Leg objects - see below for more details on the Leg object.
distance_unit *optional*	string	The distance unit can be miles (`mi`) or kilometers (`km`). If no `distance_unit` is provided the default will be set to `km` .

Leg Object

Parameter	Type	Description
departure_airport	string	The departure airport IATA code.
destination_airport	string	The destination airport IATA code.
cabin_class *optional*	string	The seating class that the passengers are taking for the leg. Allowable values are: `economy` and `premium`. If no value is provided for this param, `economy` will be set by default.

Create a Flight Estimate Request

shell

```
curl "https://www.carboninterface.com/api/v1/estimates"
  -H "Authorization: Bearer API_KEY"
  -H "Content-Type: application/json"
  -X POST
  -d {
        "type": "flight",
        "passengers": 2,
        "legs": [
          {"departure_airport": "sfo", "destination_airport": "yyz"},
          {"departure_airport": "yyz", "destination_airport": "sfo"}
        ]
      }
```

Create a Flight Estimate Response

json

```
{
  "data": {
    "id": "d60edacc-cf6c-4da7-b5de-c538de4ce5ee",
    "type": "estimate",
    "attributes": {
```

```
        "passengers": 2,
        "legs": [
          {
            "departure_airport": "SFO",
            "destination_airport": "YYZ"
          },
          {
            "departure_airport": "YYZ",
            "destination_airport": "SFO"
          }
        ],
        "estimated_at": "2020-07-24T02:25:50.837Z",
        "carbon_g": 1077098,
        "carbon_lb": 2374,
        "carbon_kg": 1077,
        "carbon_mt": 1,
        "distance_unit": "km",
        "distance_value": 7454.15
      }
    }
  }
```


Shipping

The shipping estimate request will accept shipment details and compute the carbon emissions for the shipment.

Parameter	Type	Description
type	string	`shipping`
weight_unit	string	The weight unit that is being used. This param can be set to grams (`g`), pounds (`lb`), kilograms (`kg`) or tonnes (`mt`).

Parameter	Type	Description
weight_value	decimal	The weight value of the shipment in the unit of measurement set above in the `weight_unit` param.
distance_unit	string	The distance unit can be miles (`mi`) or kilometers (`km`).
distance_value	decimal	The distance value is the distance the shipment has travelled in the unit of measurement set above in the `distance_value` param.
transport_method	string	The method the shipment is travelling. Values for this param can be `ship` , `train` , `truck` and `plane` .

Create a Shipping Estimate Request

shell

```
curl "https://www.carboninterface.com/api/v1/estimates"
  -H "Authorization: Bearer API_KEY"
  -H "Content-Type: application/json"
  -X POST
  -d {
      "type": "shipping",
      "weight_value": 200,
      "weight_unit": "g",
      "distance_value": 2000,
      "distance_unit": "km",
      "transport_method": "truck"
    }
```

Create a Shipping Estimate Response

json

```
{
  "data": {
```

```
      "id": "4746e4ba-6605-4acc-802b-fd229a9503b4",
      "type": "estimate",
      "attributes": {
        "distance_value": "2000.0",
        "distance_unit": "km",
        "weight_value": "200.0",
        "weight_unit": "g",
        "transport_method": "truck",
        "estimated_at": "2020-07-31T13:00:04.446Z",
        "carbon_g": 25,
        "carbon_lb": 0.06,
        "carbon_kg": 0.03,
        "carbon_mt": 0.0
      }
    }
  }
```


Vehicle

The vehicle estimate request will accept the vehicle and trip details to compute the carbon emissions for the trip.

Parameter	Type	Description
type	string	`vehicle`
distance_unit	string	The distance unit that is being used. This param can be set to miles (`mi`) or kilometers (`km`).
distance_value	decimal	The distance of the trip in the units specified in the `distance_unit` param.
vehicle_model_id	string	The `id` of the vehicle model which can be found via the Vehicle Makes and Vehicle Models endpoints below.

Create a Vehicle Estimate Request

shell

```shell
curl "https://www.carboninterface.com/api/v1/estimates"
  -H "Authorization: Bearer API_KEY"
  -H "Content-Type: application/json"
  -X POST
  -d {
      "type": "vehicle",
      "distance_unit": "mi",
      "distance_value": 100,
      "vehicle_model_id": "7268a9b7-17e8-4c8d-acca-57059252afe9"
    }
```

Create a Vehicle Estimate Response

json

```json
{
  "data": {
    "id": "6108d711-be04-4dc4-93f9-43d969fd5273",
    "type": "estimate",
    "attributes": {
      "distance_value": 100.0,
      "vehicle_make": "Toyota",
      "vehicle_model": "Corolla",
      "vehicle_year": 1993,
      "vehicle_model_id": "7268a9b7-17e8-4c8d-acca-57059252afe9",
      "distance_unit": "mi",
      "estimated_at": "2021-01-10T15:24:32.568Z",
      "carbon_g": 37029,
      "carbon_lb": 81.64,
      "carbon_kg": 37.03,
      "carbon_mt": 0.04
    }
  }
}
```

Vehicle Makes

HTTP Request

```
GET https://www.carboninterface.com/api/v1/vehicle_makes
```

The Vehicle Makes endpoint will return an array of Vehicle Makes that can be queried for their specific Vehicle Models.

Get the list of vehicle makes request

shell

```shell
curl "https://www.carboninterface.com/api/v1/vehicle_makes"
  -H "Authorization: Bearer API_KEY"
  -H "Content-Type: application/json"
  -X GET
```

Get the list of vehicle makes response

json

```json
[
  {
    "data": {
      "id": "4c1e16e1-7967-4394-b3cb-15f4577dffa1",
      "type": "vehicle_make",
      "attributes": {
          "name": "Ferrari",
          "number_of_models": 243
      }
    }
  },
  {
    "data": {
      "id": "c0d79b67-76e8-442d-b105-2c73501948a9",
      "type": "vehicle_make",
      "attributes": {
          "name": "Dodge",
```

```
            "number_of_models": 2608
          }
        }
      },
      {
        "data": {
          "id": "c983d62d-e823-4bfb-92a8-45d161be1fe1",
          "type": "vehicle_make",
          "attributes": {
              "name": "Subaru",
              "number_of_models": 924
          }
        }
      },
      {
        "data": {
          "id": "2b1d0cd5-59be-4010-83b3-b60c5e5342da",
          "type": "vehicle_make",
          "attributes": {
              "name": "Toyota",
              "number_of_models": 2162
          }
        }
      }
    ]
```


Vehicle Models

The Vehicle Models endpoint accepts a single parameter in the url, `vehicle_make_id` and then returns all of the Vehicle Models that belong to the specified Vehicle Make. Once you identify the Vehicle Model you want to perform a vehicle estimate on, copy the `id` from the Vehicle Model and set that as the `vehicle_model_id` in the Vehicle estimate paramters.

HTTP Request

```
GET https://www.carboninterface.com/api/v1/vehicle_make/<vehicle_make_id>/vehicl
```

URL Parameters

Parameter	Description
vehicle_make_id	The ID of the Vehicle Make

Get the list of Vehicle Models that belong to a Vehicle Make request

shell

```
curl "https://www.carboninterface.com/api/v1/vehicle_makes/2b1d0cd5-59be-
  -H "Authorization: Bearer API_KEY"
  -H "Content-Type: application/json"
  -X GET
```

Get the list of Vehicle Models that belong to a Vehicle Make response

json

```
[
  {
    "data": {
      "id": "7268a9b7-17e8-4c8d-acca-57059252afe9",
      "type": "vehicle_model",
      "attributes": {
          "name": "Corolla",
          "year": 1993,
          "vehicle_make": "Toyota"
      }
    }
  },
  {
```

```
      "data": {
        "id": "a2d97d19-14c0-4c60-870c-e734796e014e",
        "type": "vehicle_model",
        "attributes": {
            "name": "Camry",
            "year": 1993,
            "vehicle_make": "Toyota"
        }
      }
    },
    {
      "data": {
        "id": "14949244-b6d1-4a11-970f-73f75408f931",
        "type": "vehicle_model",
        "attributes": {
            "name": "Corolla Wagon",
            "year": 1993,
            "vehicle_make": "Toyota"
        }
      }
    }
  ]
```


Fuel Combustion

The fuel combustion estimate endpoint will provide carbon emissions in grams, pounds, kilograms and tonnes. To create a fuel combustion estimate, provide the following params:

Parameter	Type	Description
type	string	`fuel_combustion`

Parameter	Type	Description
fuel_source_type	string	The `api_name` of the fuel source you want to calculate combustion emissions from. The full list of fuel sources available can be found here.
fuel_source_unit	string	The `unit` of the fuel_source that you want combustion emissions from. Units are specific to the `fuel_source_type` so please view the units availabe per fuel source here.
fuel_source_value	decimal	Amount of the above specific unit and fuel source that you want an emissions estimate for.

Create a Fuel Combustion Estimate Request

shell

```
curl "https://www.carboninterface.com/api/v1/estimates"
  -H "Authorization: Bearer API_KEY"
  -H "Content-Type: application/json"
  -X POST
  -d {
        "type": "fuel_combustion",
        "fuel_source_type": "dfo",
        "fuel_source_unit": "btu",
        "fuel_source_value": 2
      }
```

Create a Fuel Combustion Estimate Response

json

```
{
  "data": {
    "id": "2d968fce-859d-4dc1-9489-987e795f42bb",
    "type": "estimate",
    "attributes": {
```

```
        "fuel_source_type": "dfo",
        "fuel_source_unit": "btu",
        "fuel_source_value": 2,
        "estimated_at": "2020-07-24T02:23:24.441Z",
        "carbon_g": 146320,
        "carbon_lb": 322.58,
        "carbon_kg": 146.32,
        "carbon_mt": 0.15
      }
    }
  }
```


Estimate

This endpoint retrieves a specific Estimate. The JSON response will return the estimate object based on the type of estimate.

Get an Estimate Request

HTTP Request

```
GET https://www.carboninterface.com/api/v1/<ID>
```

URL Parameters

Parameter	Description
ID	The ID of the Estimate to retrieve

shell

```
curl "https://www.carboninterface.com/api/v1/estimates/9b1e2132-6216-4c64
  -H "Authorization: Bearer API_KEY"
```

Get an Estimate Response

json

```
// Electricity Estimate Object
{
  "data": {
    "id": "2d968fce-859d-4dc2-9489-987e795f42bb",
    "type": "estimate",
    "attributes": {
      "country": "us",
      "state": "fl",
      "electricity_unit": "mwh",
      "electricity_value": "42.0",
      "estimated_at": "2020-07-24T02:23:24.441Z",
      "carbon_g": 18051428,
      "carbon_lb": 39796,
      "carbon_kg": 18051,
      "carbon_mt": 18
    }
  }
}

// Flight Estimate Object
{
  "data": {
    "id": "9b1e2132-7216-4c64-84b2-23cf31eb472e",
    "type": "estimate",
    "attributes": {
      "passengers": 2,
      "legs": [
        {
          "departure_airport": "SFO",
          "destination_airport": "YYZ"
        },
        {
          "departure_airport": "YYZ",
          "destination_airport": "SFO"
        }
      ],
      "estimated_at": "2020-07-28T00:52:01.373Z",
```

```
      "carbon_g": 2140866,
      "carbon_lb": 4719.8,
      "carbon_kg": 2140.87,
      "carbon_mt": 2.14
    }
  }
}

// Shipping Estimate Object
{
  "data": {
    "id": "4746e4ba-6505-4acc-802b-fd229a9503b5",
    "type": "estimate",
    "attributes": {
      "distance_value": "2000.0",
      "distance_unit": "km",
      "weight_value": "200.0",
      "weight_unit": "g",
      "transport_method": "truck",
      "estimated_at": "2020-07-31T13:00:04.446Z",
      "carbon_g": 25,
      "carbon_lb": 0.06,
      "carbon_kg": 0.03,
      "carbon_mt": 0.0
    }
  }
}

// Vehicle Estimate Object
{
  "data": {
    "id": "30e81e0b-596b-40c0-b6dd-847954e60078",
    "type": "estimate",
    "attributes": {
      "distance_value": 100.0,
      "vehicle_make": "toyota",
      "vehicle_model": "corolla",
      "vehicle_year": 2017,
      "distance_unit": "mi",
      "estimated_at": "2020-11-14T06:03:22.752Z",
```

```
        "carbon_g": 28900,
        "carbon_lb": 63.71,
        "carbon_kg": 28.9,
        "carbon_mt": 0.03
      }
    }
  }
```

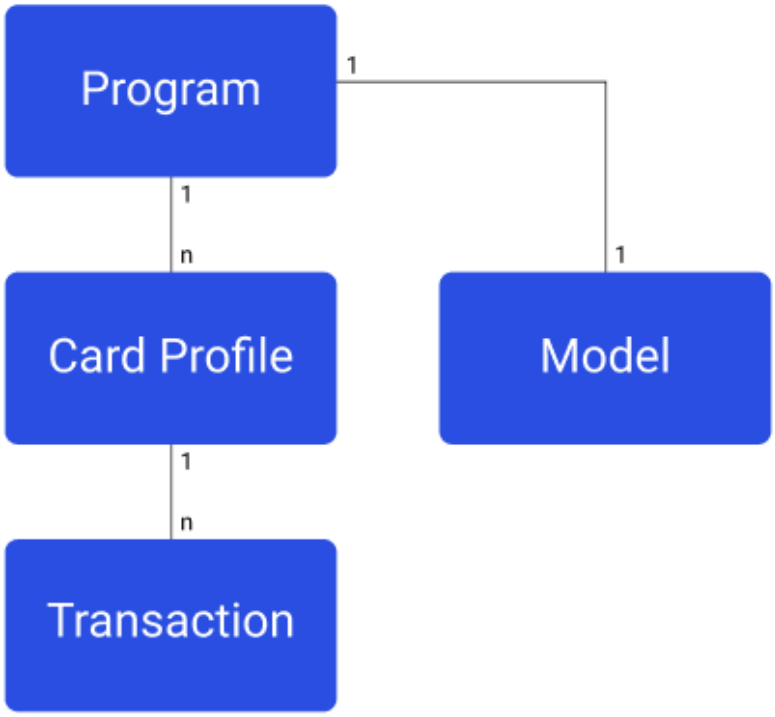
Carbon Ledger API

> Carbon Ledger is currently in beta which means the API specifications may change. If you are launching in production, please email ledger@carboninterface.com to access the production emissions model.

Carbon Ledger makes it possible to estimate the CO_2e (CO_2 equivalent) emissions of transactions that have a merchant category code and other merchant details.

Carbon Ledger is backed by a sophisticated model that can estimate the carbon emissions of any transaction based on the merchant category code.

The data model for Carbon Ledger is as follows:



Program

> You will only be able to create `test` Programs in the Carbon Ledger UI which will result in stubbed emissions estimates.

Programs are the parent entity of your Carbon Ledger API integration. Programs are created in your dashboard and require you to set the `name`, `model` and `environment` of your Carbon Ledger implementation. The Program object is how you will differentiate between test and production.

Programs cannot be created via the Carbon Ledger API.

Model

Programs require a model to be selected during the creation process. Models are the underlying algorithm that calculates the emissions for a credit card transactions. We currently have one model developed for production.

We will be sharing more about our model shortly.

Card Profile

A Card Profile represents the habits and behaviors of a spender (user of a card). We recommend that a card has a 1:1 relationship with a Card Profile. Currently, Card Profiles store the diet and transportation habits of a user. These two categories (transportation and food) account for a significant amount of spend on consumer credit cards. With these behaviors stored in a Card Profile, we can estimate CO_2e emissions with greater accuracy.

To map a Card Profile to a spender in your system easily, we have an `external_id` parameter.

Parameter	Type	Description
external_id	string	Set this to the ID of the card so general habbits can be added to improve emissions estimate accuracy
diet_habit	string	The diet of the card user. This is used to improve the estimates of spend in restaurant, grocery and snack categories. Options for this field are: `vegan`, `vegetarian`, `plant_based` and `omnivore`
transportation_method	string	The primary transportation method of the card user. This is used to improve the estimates of spend of transportation-related categories. Options for this field are: `public_transportation`, `compact_vehicle`, `midsize_vehicle` and `large_vehicle`

> A Card Profile is required to generate transaction CO_2e emissions estimates. If you don't want to include a workflow or questionnaire to collect the

> `diet_habit` and `transportation_method` data in your application, then Carbon Ledger will set `omnivore` and `midsize_vehicle` as defaults.

Create a Card Profile

Request

shell

```
curl "https://www.carboninterface.com/api/v1/carbon_ledger/programs/:prog
  -H "Authorization: Bearer API_KEY"
  -H "Content-Type: application/json"
  -X POST
  -d  {
        "external_id": "a12-b34-c56",
        "diet_habit": "omnivore",
        "transportation_method": "midsize_vehicle"
      }
```

Response

json

```
{
  "data": {
    "id": "97141aae-1142-4f27-8881-89c928f6a9d0",
    "type": "card_profile",
    "attributes": {
      "external_id": "a12-b34-c56",
      "diet_habit": "omnivore",
      "transportation_method": "midsize_vehicle"
    }
  }
}
```

Update a Card Profile

Request

shell

```
curl "https://www.carboninterface.com/api/v1/carbon_ledger/programs/:prog
  -H "Authorization: Bearer API_KEY"
  -H "Content-Type: application/json"
  -X POST
  -d  {
        "external_id": "a12-b34-c56",
        "diet_habit": "vegan",
        "transportation_method": "large_vehicle"
      }
```

Response

json

```
{
  "data": {
    "id": "97141aae-1142-4f27-8881-89c928f6a9d0",
    "type": "card_profile",
    "attributes": {
      "external_id": "a12-b34-c56",
      "diet_habit": "vegan",
      "transportation_method": "large_vehicle"
    }
  }
}
```

Get a Card Profile

Request

shell

```
curl --request GET \
  --url 'https://www.carboninterface.com/api/v1/carbon_ledger/programs/:p
  --header 'Authorization: Bearer <API KEY>' \
  --header 'Content-Type: application/json'
```

Response

```json
{
  "data": {
    "id": "97141aae-1142-4f27-8881-89c928f6a9d0",
    "type": "card_profile",
    "attributes": {
      "external_id": "a12-b34-c56",
      "diet_habit": "vegan",
      "transportation_method": "large_vehicle"
    }
  }
}
```

Get a List of Card Profiles

Request

```shell
curl --request GET \
  --url 'https://www.carboninterface.com/api/v1/carbon_ledger/programs/:p
  --header 'Authorization: Bearer <API KEY>' \
  --header 'Content-Type: application/json'
```

Response

```json
[
  {
    "data": {
      "id": "97141aae-1142-4f27-8881-89c928f6a9d0",
      "type": "card_profile",
      "attributes": {
        "external_id": "a12-b34-c56",
        "diet_habit": "omnivore",
        "transportation_method": "public_transportation"
      }
    }
  },
```

```
  {
    "data": {
      "id": "97141aae-1142-4f27-8881-89c928f6a9d0",
      "type": "card_profile",
      "attributes": {
        "external_id": "d67-e89-f10",
        "diet_habit": "vegan",
        "transportation_method": "compact_vehicle"
      }
    }
  },
  ...
]
```

Delete a Card Profile

shell

```
curl --request DELETE \
  --url 'https://www.carboninterface.com/api/v1/carbon_ledger/programs/:p
  --header 'Authorization: Bearer <API KEY>' \
  --header 'Content-Type: application/json'
```

A successul DELETE request will result in a response with a `204` status code.

Transaction

The Transaction object receives metadata about a transaction that occured and uses that metadata to calculate the CO_2e emissions. Transactions belong to a Card Profile and cannot be deleted.

> We do not store any Personally Identifiable Information (PII) on the transaction object.

Parameter	Type	Description
amount_cents	integer	Pre-tax amount of the transaction in cents
currency	string	Currency the transaction was made in. `USD` and `CAD` are supported. If a currency isn't provided, `USD` will be used as the default. If a transaction was performed in a different currency than `USD` or `CAD`, we recommend converting the transaction from its native currency to `USD` and then making the request.
external_id	string	Unique record identifier from a system outside of Carbon Ledger. In this case, this may be the ID of the transaction in your application.
merchant_category	string	Merchant Category of the merchant (where the spend is payment processing occurs)
merchant_category_code	string	Merchant Category Code (MCC) of the merchant (where the spend is payment processing occurs)
merchant_name	string	Name of the merchant (where the spend is payment processing occurs)
merchant_country	string	Country of the merchant (where the spend is payment processing occurs)
merchant_state	string	State of the merchant (where the spend is payment processing occurs)
merchant_city	string	City of the merchant (where the spend is payment processing occurs)
merchant_postal_code	string	Postal code of the merchant (where the spend is payment processing occurs)

Create a Transaction

Request

shell

```
curl "https://www.carboninterface.com/api/v1/carbon_ledger/programs/:prog
  -H "Authorization: Bearer API_KEY"
  -H "Content-Type: application/json"
  -X POST
  -d {
        "amount_cents": 10000,
        "currency": "USD",
        "external_id": "1234",
        "merchant_category": "quick_serve_restauraunt",
        "merchant_category_code": "5814",
        "merchant_name": "Philz Coffee",
        "merchant_country": "US",
        "merchant_state": "CA",
        "merchant_city": "San Francisco",
        "merchant_postal_code": "90210"
     }
```

Response

json

```
{
  "data": {
    "id": "0f7dc254-2515-4de9-9038-bb348a2c06bb",
    "type": "transaction",
    "attributes": {
      "external_id": "1234",
      "amount_cents": 10000,
      "currency": "USD",
      "carbon_grams": 100000,
      "merchant_name": "Philz Coffee",
      "merchant_category": "quick_serve_restauraunt",
      "merchant_category_code": 5814,
      "merchant_city": "San Francisco",
      "merchant_postal_code": "90210",
      "merchant_state": "CA",
```

```
      "merchant_country": "US",
      "calculated_at": "2021-04-17T20:04:28.594Z"
    },
    "relationships": {
      "merchant": {
          "data": {
              "id": "0f7dc254-2515-4de9-9038-bb348a2c06bb",
              "type": "merchant"
          }
      }
    }
  }
}
```

Get a Transaction

Request

shell

```
curl --request GET \
  --url 'https://www.carboninterface.com/api/v1/carbon_ledger/programs/:p
  --header 'Authorization: Bearer <API KEY>' \
  --header 'Content-Type: application/json'
```

Response

json

```
{
  "data": {
    "id": "0f7dc254-2515-4de9-9038-bb348a2c06bb",
    "type": "transaction",
    "attributes": {
      "external_id": "1234",
      "amount_cents": 10000,
      "currency": "USD",
      "carbon_grams": 100000,
      "merchant_name": "Philz Coffee",
      "merchant_category": "quick_serve_restauraunt",
```

```
      "merchant_category_code": 5814,
      "merchant_city": "San Francisco",
      "merchant_postal_code": "90210",
      "merchant_state": "CA",
      "merchant_country": "US",
      "calculated_at": "2021-04-17T20:04:28.594Z"
    },
    "relationships": {
      "merchant": {
        "data": {
          "id": "0f7dc254-2515-4de9-9038-bb348a2c06bb",
          "type": "merchant"
        }
      }
    }
  }
}
```


Get a List of Transactions

Request

shell

```
curl --request GET \
  --url 'https://www.carboninterface.com/api/v1/carbon_ledger/programs/:p
  --header 'Authorization: Bearer <API KEY>' \
  --header 'Content-Type: application/json'
```

Response

json


```
[
  {
    "data": {
      "id": "0f7dc254-2515-4de9-9038-bb348a2c06bb",
      "type": "transaction",
      "attributes": {
        "external_id": "1234",
```

```
        "amount_cents": 10000,
        "currency": "USD",
        "carbon_grams": 100000,
        "merchant_name": "Philz Coffee",
        "merchant_category": "quick_serve_restauraunt",
        "merchant_category_code": 5814,
        "merchant_city": "San Francisco",
        "merchant_postal_code": "90210",
        "merchant_state": "CA",
        "merchant_country": "US",
        "calculated_at": "2021-04-17T20:04:28.594Z"
      },
      "relationships": {
        "merchant": {
          "data": {
            "id": "0f7dc254-2515-4de9-9038-bb348a2c06bb",
            "type": "merchant"
          }
        }
      }
    },
    ...
]
```