# H2SCM - Supply and Demand API Documentation

Supply and Demand API

2024-02-23

# SupplyDemandAPI

## POST /repositories/{repo}/simulation

Run a simulation using the SCM store.

**Request Parameters:**

```
repo: string;  // The name of the repository.
debug?: boolean;  // To return or not to return debug output.
instances?: string;  // A comma separated list of the instance ID's to use for the simulation
    (defaults to default).
```

**Request Body:**

Content: application/json | Type: APIInput

```
{
    location: locationInput;
    fuel: fuelInput;
    query: queryInput;
}
```

**Response 200:**

OK.

Content: application/json | Type: APIResponse

```
{
    fuel?: Array<fuel>;
    logistic?: Array<logistic>;
    storageRental?: Array<storageRental>;
    matches: Array<match>;
}
```

## OPTIONS /repositories/{repo}/simulation

**Request Parameters:**

```
repo: string;  // The name of the repository.
```

**Response 200:**

Default response.

# Schemas

## locationInput

```
{
  lat: number;
  long: number;
}
```

## fuelInput

```
{
  amount: number;  // The total amount of hydrogen required per week in kg.
}
```

## instanceInput

Type: string

## queryInput

```
{
  instance: Array<instanceInput>;  // The public or private (□) instance(s) to use when querying.
}
```

## APIInput

```
{
  location: locationInput;
  fuel: fuelInput;
  query: queryInput;
}
```

## dispenser

```
{
  id: string;
  name: string;
  lat: number;
  long: number;
}
```

## productionSource

Type: string

Values: Grid, Grid Renewable, Nuclear, Wind, Solar

## producer

```
{
  id: string;
  name: string;
  weeklyProductionCapacity: number;
  productionCO2e?: number;
  source?: productionSource;
  storedIn?: Array<string>;
}
```

## service

```
{
  id: string;
  name: string;
  transportCO2e?: number;
  exclusiveDownstreamCompanies?: string;
  exclusiveUpstreamCompanies?: string;
}
```

## quote

```
{
  id: string;
  monetaryValuePerUnit: number;
  currency: string;
  unit: string;
}
```

## company

```
{
  id: string;
}
```

## graphInstance

Type: string

## fuel

```
{
  dispenser: dispenser;
  producer: producer;
  service: service;
  quote: quote;
  company: company;
  instance: graphInstance;
}
```

### vehicle

```
{
    id: string;
    name: string;
    availableQuantity: number;
    transportDistance: number;
}
```

### logistic

```
{
    service: service;
    vehicle: vehicle;
    quote: quote;
    company: company;
    instance: graphInstance;
}
```

### storage

```
{
    id: string;
    name: string;
    availableQuantity: number;
    capacity: number;
}
```

### storageRental

```
{
    service: service;
    storage: storage;
    quote: quote;
    company: company;
    instance: graphInstance;
}
```

### instance

```
{
    id: string;
    name: string;
    exclusiveDownstream: boolean;
    exclusiveUpstream: boolean;
    type?: string;
    instance: graphInstance;
}
```

### breakdown

```
{
  serviceType: string;
  service: string;  // The ID of the service that is attached to this breakdown.
  quantity: number;
  perUnit: number;
  unit: string;
  value: string;
}
```

### cost

```
{
  total: number;
  breakdown: Array<breakdown>;
}
```

### CO2e

```
{
  total: number;
  breakdown: Array<breakdown>;
}
```

### productionMethod

Type: string

Values: ElectrolyticHydrogen, SteamMethaneReformingHydrogen, Hydrogen

### productionCapacity

```
{
  weekly: number;
  weeklyUsed: number;
}
```

### location

```
{
  lat: number;
  long: number;
}
```

### production

```
{
  method: productionMethod;
  source?: productionSource;
  capacity: productionCapacity;
  location: location;
}
```

## match

```
{
    fuel: instance;
    logistic: instance;
    storage: instance;
    cost: cost;
    CO2e?: CO2e;
    production: production;
    transportDistance: number;
}
```

## APIResponse

```
{
    fuel?: Array<fuel>;
    logistic?: Array<logistic>;
    storageRental?: Array<storageRental>;
    matches: Array<match>;
}
```