
isoprene-pumpjack Documentation

Release 0.0.1

Tom Milligan

Mar 31, 2017

Contents:

1	Installation	1
1.1	Use	1
2	Neo-D3-JSON	3
2.1	Neo-D3-JSON Specification	3
2.2	Neo-D3-JSON Example Response	4
3	RESTful API	7
3.1	General API Notes	7
3.2	Neat API	7
3.3	Full API	9
4	Indices and tables	13
	HTTP Routing Table	15

CHAPTER 1

Installation

Install with pip:

```
pip install isoprene-pumpjack
```

Use

To run with Gunicorn prod server:

```
gunicorn isoprene_pumpjack.wsgi:app
```

To run with Flask dev server:

```
isoprene-pumpjack
```


Data is often returned from isoprene-pumpjack to instruct synpatic-scout how to draw a graph.

This is done in a standard JSON format, described below:

Neo-D3-JSON Specification

Nodes & Links

The object contains an array of **nodes** and an array of **links**. This is a naming convention taken from d3 force-constrained graphs:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "nodes": [],
  "links": []
}
```

The response from Neo is deduplicated, so each node and link is unique.

Both nodes and links have the following standard properties:

```
{
  "labels": string[],      # Array of Neo labels
  "id": any,               # Neo internal id (not reccomended for external use)
  "props": {},             # Object containing any additional properties
  ...
}
```

Nodes

Node objects have no additional properties.

Links

Link objects also contain the following properties:

```
{
  "target": any,          # Target of link
  "source": any,          # Source of link
  ...
}
```

`source` and `target` will match the `id` property of a node object.

All links are considered directed by neo - it is up to the UI to render nondirected links if required.

Please note: unlike Neo, link labels are represented as an array of length 1, in order to be consistent with node properties.

Neo-D3-JSON Example Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "nodes": [
    {
      "labels": [
        "Document"
      ],
      "id": 0,
      "props": {
        "isopump_fully_loaded": true,
        "label": "AVrDasjxocluSgWq6vsT",
        "isopump_load_last": 1490949576577,
        "id": "AVrDasjxocluSgWq6vsT",
        "isopump_load_initial": 1490714160644
      }
    },
    {
      "labels": [
        "Dolphin"
      ],
      "id": 1,
      "props": {
        "isopump_fully_loaded": true,
        "id": "Zap",
        "label": "Zap",
        "isopump_load_last": 1490949577244,
        "isopump_load_initial": 1490714160644
      }
    }
  ],
}
```

```
"links": [
  {
    "props": {
      },
    "labels": [
      "Source"
    ],
    "id": 0,
    "target": 0,
    "source": 1
  }
]
```


General API Notes

If one of the backend services (Neo4j, Elasticsearch) is unavailable, a 503 error will be returned instead of a JSON response:

statuscode 503 Service unavailable

Neat API

This API documentation is manually updated.

GET /dev/elastic/set/dolphins

Create and load dolphins index

Status Codes

- 201 Created – Document index created

GET /dev/neo/set/dolphins

Drop database and upload dolphins JSON data into Neo4j

Status Codes

- 201 Created – Neo4j graph loaded

GET /dev/elastic/search

Return the elastic results of a document search using query parameters.

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
```

```
{
  "_index": "dolphins",
  "_source": {
    "dolphins": [
      "Zap",
      "CCL"
    ]
  },
  "_id": "AVrDasjxocluSgWq6vsT",
  "_type": "dolphin_sighting",
  "_score": 2.6732156
},
{
  "_index": "dolphins",
  "_source": {
    "dolphins": [
      "Zap",
      "Double"
    ]
  },
  "_id": "AVrDasj8ocluSgWq6vsU",
  "_type": "dolphin_sighting",
  "_score": 2.6732156
}
]
```

Query Parameters

- **label** (*string*) – id of dolphin to search documents for

Status Codes

- 200 OK – OK

GET /dev/elastic/reset

Drop elastic indexes

Status Codes

- 204 No Content – Indexes dropped

GET /dev/elastic/seed

Seed a node (and mentioning documents) from elastic to neo, and return the node id

Example response:

```
HTTP/1.1 201 CREATED
Content-Type: application/json

{
  "id": "Zap"
}
```

Query Parameters

- **label** (*string*) – id of dolphin to search documents for

Status Codes

- 201 Created – Created graph node(s)

GET `/dev/neo/fullgraph`

Get JSON representing full graph

See *Neo-D3-JSON Example Response*.

Status Codes

- 200 OK – OK

GET `/dev/neo/reset`

Drop neo4j database

Status Codes

- 204 No Content – Database dropped

GET `/configuration/synaptic-scout`

Get JSON config for synaptic-scout

GET `/`

Get JSON representing endpoints

GET `/dev/neo/subgraph/` (string: *central_node_id*)

Get JSON representing subgraph centered on a single node

See *Neo-D3-JSON Example Response*.

Status Codes

- 200 OK – OK

GET `/explore/subgraph/` (string: *central_node_id*)

Get JSON representing subgraph centered on a single node

See *Neo-D3-JSON Example Response*.

Status Codes

- 200 OK – OK

GET `/static/` (path: *filename*)

Function used internally to send static files from the static folder to the browser.

New in version 0.5.

Full API

This API documentation is automatically generated.

GET `/`

Get JSON representing endpoints

GET `/configuration/synaptic-scout`

Get JSON config for synaptic-scout

GET `/dev/elastic/reset`

Drop elastic indexes

Status Codes

- 204 No Content – Indexes dropped

GET /dev/elastic/search

Return the elastic results of a document search using query parameters.

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "_index": "dolphins",
    "_source": {
      "dolphins": [
        "Zap",
        "CCL"
      ]
    },
    "_id": "AVrDasjxocluSgWq6vsT",
    "_type": "dolphin_sighting",
    "_score": 2.6732156
  },
  {
    "_index": "dolphins",
    "_source": {
      "dolphins": [
        "Zap",
        "Double"
      ]
    },
    "_id": "AVrDasj8ocluSgWq6vsU",
    "_type": "dolphin_sighting",
    "_score": 2.6732156
  }
]
```

Query Parameters

- **label** (*string*) – id of dolphin to search documents for

Status Codes

- 200 OK – OK

GET /dev/elastic/seed

Seed a node (and mentioning documents) from elastic to neo, and return the node id

Example response:

```
HTTP/1.1 201 CREATED
Content-Type: application/json

{
  "id": "Zap"
}
```

Query Parameters

- **label** (*string*) – id of dolphin to search documents for

Status Codes

- 201 Created – Created graph node(s)

GET /dev/elastic/set/dolphins

Create and load dolphins index

Status Codes

- 201 Created – Document index created

GET /dev/neo/fullgraph

Get JSON representing full graph

See *Neo-D3-JSON Example Response*.

Status Codes

- 200 OK – OK

GET /dev/neo/reset

Drop neo4j database

Status Codes

- 204 No Content – Database dropped

GET /dev/neo/set/dolphins

Drop database and upload dolphins JSON data into Neo4j

Status Codes

- 201 Created – Neo4j graph loaded

GET /dev/neo/subgraph/ (string: *central_node_id*)

Get JSON representing subgraph centered on a single node

See *Neo-D3-JSON Example Response*.

Status Codes

- 200 OK – OK

GET /explore/subgraph/ (string: *central_node_id*)

Get JSON representing subgraph centered on a single node

See *Neo-D3-JSON Example Response*.

Status Codes

- 200 OK – OK

GET /static/ (path: *filename*)

Function used internally to send static files from the static folder to the browser.

New in version 0.5.

CHAPTER 4

Indices and tables

- `genindex`
- `modindex`
- `search`

HTTP Routing Table

/

GET /,9

/configuration

GET /configuration/synaptic-scout,9

/dev

GET /dev/elastic/reset,8

GET /dev/elastic/search,7

GET /dev/elastic/seed,8

GET /dev/elastic/set/dolphins,7

GET /dev/neo/fullgraph,9

GET /dev/neo/reset,9

GET /dev/neo/set/dolphins,7

GET /dev/neo/subgraph/(string:central_node_id),
9

/explore

GET /explore/subgraph/(string:central_node_id),
9

/static

GET /static/(path:filename),9