Census Engine RESTful API: In Requests

Note: Some requests are added to a queue, this kind of requests may take longer so they just return an `acknowledged` *status with a string token for future reference, they are processed in a "first come first served" queue. Future error or success reports may follow the* `acknowledged` *response in the form of an "out request" report, which is an HTTP request from Census Engine to Census Control. One of the main reasons of doing this is that many operations must be made sequentially and Census Control should not wait for them to be done, just receive a report when the request is finally done.*

Note: If Census Engine can't send messages to Census Control every out request will just log `WARNING: Unreachable Census Control server.`

POST /control

Registers the server information of the Census Control master, used by the `CensusControl` module to send HTTP messages to Census Control. This is the first thing that Census Control must request before anything else.

Request data

`host` : The Census Control host name.
`port` : The Census Control port.

```
{
  host: "census-control",
  port: 9595
}
```

Response data

`status` : Value should be `success` , which means the Census Control server information is now registered to this Census Engine instance, if json was invalid value should be `badrequest` .

Response status code `200` :

```
{
  status: "success"
```

```
}
```

Response status code `400`:

```
{
  status: "bad request",
  errors: [
    "'host' field missing.",
    "'port' field missing."
  ]
}
```

Response status code `500`:

```
{
  status: "unreachable host"
}
```

GET /control

Returns the server information of the Census Control server that is registered on this instance.

Request data

```
No data needed.
```

Response data

`host`: The Census Control host name.
`port`: The Census Control port.

Response status code `200`:

```
{
  host: "census-control",
  port: 9595
}
```

POST /graph

Note: This request is added to the main queue.

Note: Every imported node needs an `id` attribute with a string value in Neo4j.

Imports a graph from a Neo4j database, has to be formated for a specific algorithm. This action will automatically delete any previous graph imported to Census Engine.

Request data

`token` : A unique string id for the request, this is used to identify future reports.
`algorithm` : The algorithm for which will be formated the graph.
`tag` : The Neo4j tag that will be used to import the nodes.
`host` : The Neo4j hostname.
`port` : The Neo4j port.
`user` *Optional*: The desired user for the Census Engine instance to use.
`password` *Optional*: The password of the Census Engine user.

```
{
  token: "AJD23JS391",
  algorithm: "SSCloseness",
  tag: "Person",
  host: "10.3.10.123",
  port: 7474,
  user: "root",
  password: "root"
}
```

Response data

`status` : Value should be `acknowledged` , which only means the request was enqueued to the Census Engine instance. Future out requests may report errors.

Response status code `200` :

```
{
  status: "acknowledged"
}
```

Response status code `400` :

```
{
  status: "bad request",
  errors: [
    "'token' field missing.",
    "No such algorithm 'InexistentAlgo'",
    "'algorithm' field missing.",
    "'tag' field missing.",
    "'host' field missing.",
    "'port' field missing."
```

```
    ]
}
```

POST /compute

Note: This request is added to the main queue.

Enqueues a graph computation, must match the current graph format. The request needs a string token generated by Census Control, which is used to reference the request on reports of errors or success.

Request data

`token`: A unique string id for the request, this is used to identify future reports.
`algorithm`: The name of the algorithm that will be computed, must match the current graph format.
`timeCreation`: The Unix milliseconds time stamp when the request was created.
`vars`: All the variables needed for the computation of the algorithm, this depend on each algorithm needs.

```
{
  token: "AJD23JSA941",
  algorithm: "SSCloseness"
  creationTime: 1394231356274,
  vars: {
    source: 91
  }
}
```

Response data

`status`: Value should be `acknowledged`, which only means the request was enqueued to the Census Engine instance. Future out requests may report errors.

Response status code 200:

```
{
  status: "acknowledged"
}
```

Response status code `400`:

```
{
  status: "bad request",
  errors: [
    "'token' field missing.",
```

```
    "No such algorithm 'InexistentAlgo'",
    "'algorithm' field missing.",
    "'timeCreation' field missing.",
  ]
}
```

Census Engine RESTful API: Error and Success Reports

After registering a Census Control web hook, reports will be sent to the following paths:

POST /censusengine/report

Graph import finished:

```
{
  token: "asd124-asdf-12351af-214",
  status: "finished"
}
```

Computation finished:

```
{
  token: "asd124-asdf-12351af-214",
  status: "finished"
}
```

POST /censusengine/error

Error reports on a POST /graph

Unreachable Neo4j server (When importing the graph):

```
{
  token: "asd124-asdf-12351af-214",
  status: "error",
  error: "unreachable-neo4j",
  on: "graph-import"
}
```

Invalid Neo4j format:

```
{
  token: "asd124-asdf-12351af-214",
  status: "error",
  error: "invalid-neo4j-format",
  on: "graph-import"
}
```

Error reports on a POST /compute

Unreachable Neo4j server (When inserting back the result of a computation):

```
{
  token: "asd124-asdf-12351af-214",
  status: "error",
  error: "unreachable-neo4j",
  on: "compute"
}
```

Computation not ready (No graph was imported):

```
{
  token: "asd124-asdf-12351af-214",
  status: "error",
  error: "missing-graph",
  on: "compute"
}
```

Computation bad request (Invalid variables for the computation):

```
{
  token: "asd124-asdf-12351af-214",
  status: "error",
  error: "invalid-variables",
  on: "compute"
}
```