The Neo4j REST API Documentation v3.5

Table of Contents

The API described in this manual been deprecated and will be removed in Neo4j 4.0.

> The API described in this manual been deprecated and will be removed in Neo4j 4.0. Cypher and procedures should be used instead, either via the HTTP API, or via Bolt using the official drivers.

The Neo4j REST API is designed with discoverability in mind, so that you can start with a `GET` on the Service root and from there discover URIs to perform other requests. While the examples below use correct URIs best practice is to *discover URIs where possible*, rather than relying on the layout in these examples. This allows for handling changes to the URI structure gracefully.

The default representation is json *(http://www.json.org/)*, both for responses and for data sent with `POST`/`PUT` requests.

To interact with the JSON interface you must explicitly set the request header `Accept:application/json` for those requests that responds with data. You should also set the header `Content-Type:application/json` if your request sends data, for example when you're creating a relationship. The examples include the relevant request and response headers.

The server supports streaming results, with better performance and lower memory overhead. See Streaming for more information.

Service root

Get service root

The service root is your starting point to discover the REST API. It contains the basic starting points for the database, and some version and extension information.

Figure 1. Final Graph

Example request

- **GET** http://localhost:7474/db/data/
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "extensions" : { },
  "node" : "http://localhost:7474/db/data/node",
  "relationship" : "http://localhost:7474/db/data/relationship",
  "node_index" : "http://localhost:7474/db/data/index/node",
  "relationship_index" : "http://localhost:7474/db/data/index/relationship",
  "extensions_info" : "http://localhost:7474/db/data/ext",
  "relationship_types" : "http://localhost:7474/db/data/relationship/types",
  "batch" : "http://localhost:7474/db/data/batch",
  "cypher" : "http://localhost:7474/db/data/cypher",
  "indexes" : "http://localhost:7474/db/data/schema/index",
  "constraints" : "http://localhost:7474/db/data/schema/constraint",
  "transaction" : "http://localhost:7474/db/data/transaction",
  "node_labels" : "http://localhost:7474/db/data/labels",
  "neo4j_version" : "3.4.2-SNAPSHOT"
}
```

Streaming

All responses from the REST API can be transmitted as JSON streams, resulting in better performance and lower memory overhead on the server side. To use streaming, supply the header `X-Stream: true` with each request.

Example request

- **GET** http://localhost:7474/db/data/
- **Accept:** application/json
- **X-Stream:** true

Example response

- **200:** OK
- **Content-Type:** application/json; charset=UTF-8; stream=true

```
{
  "extensions" : { },
  "node" : "http://localhost:7474/db/data/node",
  "relationship" : "http://localhost:7474/db/data/relationship",
  "node_index" : "http://localhost:7474/db/data/index/node",
  "relationship_index" : "http://localhost:7474/db/data/index/relationship",
  "extensions_info" : "http://localhost:7474/db/data/ext",
  "relationship_types" : "http://localhost:7474/db/data/relationship/types",
  "batch" : "http://localhost:7474/db/data/batch",
  "cypher" : "http://localhost:7474/db/data/cypher",
  "indexes" : "http://localhost:7474/db/data/schema/index",
  "constraints" : "http://localhost:7474/db/data/schema/constraint",
  "transaction" : "http://localhost:7474/db/data/transaction",
  "node_labels" : "http://localhost:7474/db/data/labels",
  "neo4j_version" : "3.4.2-SNAPSHOT"
}
```

Legacy Cypher HTTP endpoint

This endpoint is deprecated. Please transition to using the transactional HTTP API (see the HTTP API Docs). Among other things it allows you to run multiple Cypher statements in the same transaction.

The Neo4j REST API allows querying with Cypher (see the Cypher documentation). The results are returned as a list of string headers (`columns`), and a `data` part, consisting of a list of all rows. Every row consists of a list of REST representations of the field value — `Node`, `Relationship`, `Path`, or any simple value like `String`.

In order to speed up queries in repeated scenarios, try not to use literals but replace them with parameters wherever possible in order to let the server cache query plans, see Use parameters for details. Also see the Cypher documentation for where parameters can be used.

Use parameters

Cypher supports queries with parameters which are submitted as JSON.

```
MATCH (x { name: { startName }})-[r]-(friend)
WHERE friend.name = { name }
RETURN TYPE(r)
```

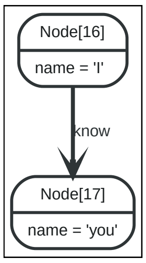


Figure 2. Final Graph

Example request

- **POST** http://localhost:7474/db/data/cypher
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "query" : "MATCH (x {name: {startName}})-[r]-(friend) WHERE friend.name = {name} RETURN TYPE(r)",
  "params" : {
    "startName" : "I",
    "name" : "you"
  }
}
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "columns" : [ "TYPE(r)" ],
  "data" : [ [ "know" ] ]
}
```

Create a node

Create a node with a label and a property using Cypher. See the request for the parameter sent with the query.

```
CREATE (n:Person { name : { name }})
RETURN n
```

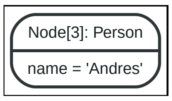


Figure 3. Final Graph

Example request

- **POST** http://localhost:7474/db/data/cypher
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "query" : "CREATE (n:Person { name : {name} }) RETURN n",
  "params" : {
    "name" : "Andres"
  }
}
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "columns" : [ "n" ],
  "data" : [ [ {
    "metadata" : {
      "id" : 3,
      "labels" : [ "Person" ]
    },
    "data" : {
      "name" : "Andres"
    },
    "paged_traverse" :
"http://localhost:7474/db/data/node/3/paged/traverse/{returnType}{?pageSize,leaseTime}",
    "outgoing_relationships" : "http://localhost:7474/db/data/node/3/relationships/out",
    "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/3/relationships/out/{-
list|&|types}",
    "labels" : "http://localhost:7474/db/data/node/3/labels",
    "create_relationship" : "http://localhost:7474/db/data/node/3/relationships",
    "traverse" : "http://localhost:7474/db/data/node/3/traverse/{returnType}",
    "extensions" : { },
    "all_relationships" : "http://localhost:7474/db/data/node/3/relationships/all",
    "all_typed_relationships" : "http://localhost:7474/db/data/node/3/relationships/all/{-list|&|types}",
    "property" : "http://localhost:7474/db/data/node/3/properties/{key}",
    "self" : "http://localhost:7474/db/data/node/3",
    "incoming_relationships" : "http://localhost:7474/db/data/node/3/relationships/in",
    "properties" : "http://localhost:7474/db/data/node/3/properties",
    "incoming_typed_relationships" : "http://localhost:7474/db/data/node/3/relationships/in/{-
list|&|types}"
  } ] ]
}
```

Create a node with multiple properties

Create a node with a label and multiple properties using Cypher. See the request for the parameter sent with the query.

```
CREATE (n:Person { props })
RETURN n
```

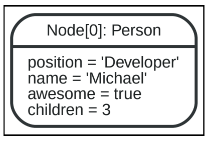


Figure 4. Final Graph

Example request

- **POST** http://localhost:7474/db/data/cypher
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "query" : "CREATE (n:Person { props } ) RETURN n",
  "params" : {
    "props" : {
      "position" : "Developer",
      "name" : "Michael",
      "awesome" : true,
      "children" : 3
    }
  }
}
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "columns" : [ "n" ],
  "data" : [ [ {
    "metadata" : {
      "id" : 0,
      "labels" : [ "Person" ]
    },
    "data" : {
      "awesome" : true,
      "children" : 3,
      "name" : "Michael",
      "position" : "Developer"
    },
    "paged_traverse" :
"http://localhost:7474/db/data/node/0/paged/traverse/{returnType}{?pageSize,leaseTime}",
    "outgoing_relationships" : "http://localhost:7474/db/data/node/0/relationships/out",
    "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/0/relationships/out/{-
list|&|types}",
    "labels" : "http://localhost:7474/db/data/node/0/labels",
    "create_relationship" : "http://localhost:7474/db/data/node/0/relationships",
    "traverse" : "http://localhost:7474/db/data/node/0/traverse/{returnType}",
    "extensions" : { },
    "all_relationships" : "http://localhost:7474/db/data/node/0/relationships/all",
    "all_typed_relationships" : "http://localhost:7474/db/data/node/0/relationships/all/{-list|&|types}",
    "property" : "http://localhost:7474/db/data/node/0/properties/{key}",
    "self" : "http://localhost:7474/db/data/node/0",
    "incoming_relationships" : "http://localhost:7474/db/data/node/0/relationships/in",
    "properties" : "http://localhost:7474/db/data/node/0/properties",
    "incoming_typed_relationships" : "http://localhost:7474/db/data/node/0/relationships/in/{-
list|&|types}"
  } ] ]
}
```

Create multiple nodes with properties

Create multiple nodes with properties using Cypher. See the request for the parameter sent with the query.

```
UNWIND { props } AS properties
CREATE (n:Person)
SET n = properties
RETURN n
```

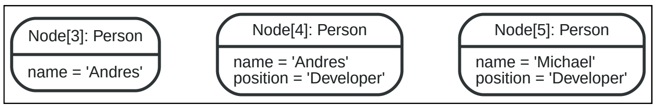


Figure 5. Final Graph

Example request

- **POST** http://localhost:7474/db/data/cypher
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "query" : "UNWIND {props} AS properties CREATE (n:Person) SET n = properties RETURN n",
  "params" : {
    "props" : [ {
      "name" : "Andres",
      "position" : "Developer"
    }, {
      "name" : "Michael",
      "position" : "Developer"
    } ]
  }
}
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "columns" : [ "n" ],
  "data" : [ [ {
    "metadata" : {
      "id" : 4,
      "labels" : [ "Person" ]
    },
    "data" : {
      "name" : "Andres",
      "position" : "Developer"
    },
    "paged_traverse" :
"http://localhost:7474/db/data/node/4/paged/traverse/{returnType}{?pageSize,leaseTime}",
    "outgoing_relationships" : "http://localhost:7474/db/data/node/4/relationships/out",
    "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/4/relationships/out/{-
list|&|types}",
    "labels" : "http://localhost:7474/db/data/node/4/labels",
    "create_relationship" : "http://localhost:7474/db/data/node/4/relationships",
    "traverse" : "http://localhost:7474/db/data/node/4/traverse/{returnType}",
    "extensions" : { },
    "all_relationships" : "http://localhost:7474/db/data/node/4/relationships/all",
    "all_typed_relationships" : "http://localhost:7474/db/data/node/4/relationships/all/{-list|&|types}",
    "property" : "http://localhost:7474/db/data/node/4/properties/{key}",
    "self" : "http://localhost:7474/db/data/node/4",
    "incoming_relationships" : "http://localhost:7474/db/data/node/4/relationships/in",
    "properties" : "http://localhost:7474/db/data/node/4/properties",
    "incoming_typed_relationships" : "http://localhost:7474/db/data/node/4/relationships/in/{-
list|&|types}"
  } ], [ {
    "metadata" : {
      "id" : 5,
      "labels" : [ "Person" ]
    },
    "data" : {
      "name" : "Michael",
      "position" : "Developer"
    },
    "paged_traverse" :
"http://localhost:7474/db/data/node/5/paged/traverse/{returnType}{?pageSize,leaseTime}",
    "outgoing_relationships" : "http://localhost:7474/db/data/node/5/relationships/out",
    "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/5/relationships/out/{-
list|&|types}",
    "labels" : "http://localhost:7474/db/data/node/5/labels",
    "create_relationship" : "http://localhost:7474/db/data/node/5/relationships",
    "traverse" : "http://localhost:7474/db/data/node/5/traverse/{returnType}",
    "extensions" : { },
    "all_relationships" : "http://localhost:7474/db/data/node/5/relationships/all",
    "all_typed_relationships" : "http://localhost:7474/db/data/node/5/relationships/all/{-list|&|types}",
    "property" : "http://localhost:7474/db/data/node/5/properties/{key}",
    "self" : "http://localhost:7474/db/data/node/5",
    "incoming_relationships" : "http://localhost:7474/db/data/node/5/relationships/in",
    "properties" : "http://localhost:7474/db/data/node/5/properties",
    "incoming_typed_relationships" : "http://localhost:7474/db/data/node/5/relationships/in/{-
list|&|types}"
  } ] ]
}
```

Set all properties on a node using Cypher

Set all properties on a node.

```
CREATE (n:Person { name: 'this property is to be deleted' })
SET n = { props }
RETURN n
```

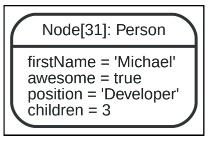


Figure 6. Final Graph

Example request

- **POST** http://localhost:7474/db/data/cypher
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "query" : "CREATE (n:Person { name: 'this property is to be deleted' } ) SET n = { props } RETURN n",
  "params" : {
    "props" : {
      "position" : "Developer",
      "firstName" : "Michael",
      "awesome" : true,
      "children" : 3
    }
  }
}
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "columns" : [ "n" ],
  "data" : [ [ {
    "metadata" : {
      "id" : 31,
      "labels" : [ "Person" ]
    },
    "data" : {
      "awesome" : true,
      "firstName" : "Michael",
      "children" : 3,
      "position" : "Developer"
    },
    "paged_traverse" :
"http://localhost:7474/db/data/node/31/paged/traverse/{returnType}{?pageSize,leaseTime}",
    "outgoing_relationships" : "http://localhost:7474/db/data/node/31/relationships/out",
    "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/31/relationships/out/{-
list|&|types}",
    "labels" : "http://localhost:7474/db/data/node/31/labels",
    "create_relationship" : "http://localhost:7474/db/data/node/31/relationships",
    "traverse" : "http://localhost:7474/db/data/node/31/traverse/{returnType}",
    "extensions" : { },
    "all_relationships" : "http://localhost:7474/db/data/node/31/relationships/all",
    "all_typed_relationships" : "http://localhost:7474/db/data/node/31/relationships/all/{-list|&|types}",
    "property" : "http://localhost:7474/db/data/node/31/properties/{key}",
    "self" : "http://localhost:7474/db/data/node/31",
    "incoming_relationships" : "http://localhost:7474/db/data/node/31/relationships/in",
    "properties" : "http://localhost:7474/db/data/node/31/properties",
    "incoming_typed_relationships" : "http://localhost:7474/db/data/node/31/relationships/in/{-
list|&|types}"
  } ] ]
}
```

Send a query

A simple query returning all nodes connected to some node, returning the node and the name property, if it exists, otherwise `NULL`:

```
MATCH (x { name: 'I' })-[r]->(n)
RETURN type(r), n.name, n.age
```

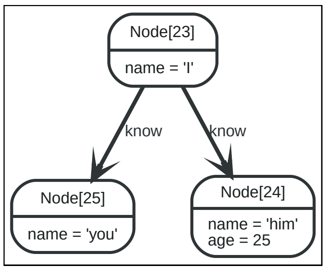


Figure 7. Final Graph

Example request

- **POST** http://localhost:7474/db/data/cypher
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "query" : "MATCH (x {name: 'I'})-[r]->(n) RETURN type(r), n.name, n.age",
  "params" : { }
}
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "columns" : [ "type(r)", "n.name", "n.age" ],
  "data" : [ [ "know", "you", null ], [ "know", "him", 25 ] ]
}
```

Return paths

Paths can be returned just like other return types.

```
MATCH path =(x { name: 'I' })--(friend)
RETURN path, friend.name
```

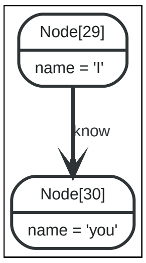


Figure 8. Final Graph

Example request

- **POST** http://localhost:7474/db/data/cypher
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "query" : "MATCH path = (x {name: 'I'})--(friend) RETURN path, friend.name",
  "params" : { }
}
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "columns" : [ "path", "friend.name" ],
  "data" : [ [ {
    "relationships" : [ "http://localhost:7474/db/data/relationship/13" ],
    "nodes" : [ "http://localhost:7474/db/data/node/29", "http://localhost:7474/db/data/node/30" ],
    "directions" : [ "->" ],
    "start" : "http://localhost:7474/db/data/node/29",
    "length" : 1,
    "end" : "http://localhost:7474/db/data/node/30"
  }, "you" ] ]
}
```

Nested results

When sending queries that return nested results like list and maps, these will get serialized into nested JSON representations according to their types.

```
MATCH (n)
WHERE n.name IN ['I', 'you']
RETURN collect(n.name)
```

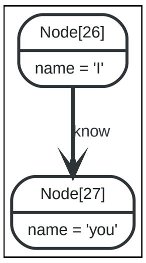


Figure 9. Final Graph

Example request

- **POST** http://localhost:7474/db/data/cypher
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "query" : "MATCH (n) WHERE n.name in ['I', 'you'] RETURN collect(n.name)",
  "params" : { }
}
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "columns" : [ "collect(n.name)" ],
  "data" : [ [ [ "I", "you" ] ] ]
}
```

Retrieve query metadata

By passing in an additional GET parameter when you execute Cypher queries, metadata about the query will be returned, such as how many labels were added or removed by the query.

```
MATCH (n { name: 'I' })
SET n:Actor
REMOVE n:Director
RETURN labels(n)
```

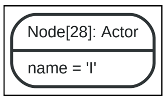


Figure 10. Final Graph

Example request

- **POST** http://localhost:7474/db/data/cypher?includeStats=true
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "query" : "MATCH (n {name: 'I'}) SET n:Actor REMOVE n:Director RETURN labels(n)",
  "params" : { }
}
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "columns" : [ "labels(n)" ],
  "data" : [ [ [ "Actor" ] ] ],
  "stats" : {
    "nodes_deleted" : 0,
    "relationship_deleted" : 0,
    "nodes_created" : 0,
    "labels_added" : 1,
    "relationships_created" : 0,
    "indexes_added" : 0,
    "properties_set" : 0,
    "contains_updates" : true,
    "indexes_removed" : 0,
    "constraints_added" : 0,
    "labels_removed" : 1,
    "constraints_removed" : 0
  }
}
```

Errors

Errors on the server will be reported as a JSON-formatted message, exception name and stacktrace.

```
MATCH (x { name: 'I' })
RETURN x.dummy/0
```

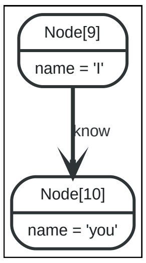


Figure 11. Final Graph

Example request

- **POST** http://localhost:7474/db/data/cypher
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "query" : "MATCH (x {name: 'I'}) RETURN x.dummy/0",
  "params" : { }
}
```

Example response

- **400:** Bad Request
- **Content-Type:** application/json;charset=utf-8

```
{
  "message": "/ by zero",
  "exception": "BadInputException",
  "fullname": "org.neo4j.server.rest.repr.BadInputException",
  "stackTrace": [

"org.neo4j.server.rest.repr.RepresentationExceptionHandlingIterable.exceptionOnNext(RepresentationExceptio
nHandlingIterable.java:48)",
    "org.neo4j.helpers.collection.ExceptionHandlingIterable$1.next(ExceptionHandlingIterable.java:76)",
    "org.neo4j.helpers.collection.IteratorWrapper.next(IteratorWrapper.java:49)",
    "org.neo4j.server.rest.repr.ListRepresentation.serialize(ListRepresentation.java:64)",
    "org.neo4j.server.rest.repr.Serializer.serialize(Serializer.java:78)",
    "org.neo4j.server.rest.repr.MappingSerializer.putList(MappingSerializer.java:66)",
    "org.neo4j.server.rest.repr.CypherResultRepresentation.serialize(CypherResultRepresentation.java:58)",
    "org.neo4j.server.rest.repr.MappingRepresentation.serialize(MappingRepresentation.java:41)",
    "org.neo4j.server.rest.repr.OutputFormat.assemble(OutputFormat.java:232)",
    "org.neo4j.server.rest.repr.OutputFormat.formatRepresentation(OutputFormat.java:172)",
    "org.neo4j.server.rest.repr.OutputFormat.response(OutputFormat.java:155)",
    "org.neo4j.server.rest.repr.OutputFormat.ok(OutputFormat.java:70)",
    "org.neo4j.server.rest.web.CypherService.cypher(CypherService.java:140)",
    "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
    "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
    "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
    "java.lang.Thread.run(Thread.java:748)"
  ],
  "cause": {
    "exception": "QueryExecutionException",
    "cause": {
      "exception": "QueryExecutionKernelException",
      "cause": {
        "exception": "ArithmeticException",
        "cause": {
          "exception": "ArithmeticException",
          "fullname": "org.neo4j.cypher.internal.util.v3_4.ArithmeticException",
          "stackTrace": [

"org.neo4j.cypher.internal.runtime.interpreted.commands.expressions.Divide.apply(Divide.scala:38)",

"org.neo4j.cypher.internal.runtime.interpreted.pipes.ProjectionPipe$$anonfun$internalCreateResults$1$$anon
fun$apply$1.apply(ProjectionPipe.scala:36)",

"org.neo4j.cypher.internal.runtime.interpreted.pipes.ProjectionPipe$$anonfun$internalCreateResults$1$$anon
fun$apply$1.apply(ProjectionPipe.scala:34)",
            "scala.collection.immutable.Map$Map1.foreach(Map.scala:116)",

"org.neo4j.cypher.internal.runtime.interpreted.pipes.ProjectionPipe$$anonfun$internalCreateResults$1.apply
(ProjectionPipe.scala:34)",

"org.neo4j.cypher.internal.runtime.interpreted.pipes.ProjectionPipe$$anonfun$internalCreateResults$1.apply
(ProjectionPipe.scala:33)",
            "scala.collection.Iterator$$anon$11.next(Iterator.scala:410)",
            "scala.collection.Iterator$$anon$11.next(Iterator.scala:410)",

"org.neo4j.cypher.internal.compatibility.v3_4.runtime.ClosingIterator.next(ResultIterator.scala:74)",

"org.neo4j.cypher.internal.compatibility.v3_4.runtime.ClosingIterator.next(ResultIterator.scala:48)",

"org.neo4j.cypher.internal.compatibility.v3_4.runtime.PipeExecutionResult$$anon$2.next(PipeExecutionResult
.scala:72)",

"org.neo4j.cypher.internal.compatibility.v3_4.runtime.PipeExecutionResult$$anon$2.next(PipeExecutionResult
.scala:70)",

"org.neo4j.cypher.internal.compatibility.ClosingExecutionResult$$anon$2$$anonfun$next$1.apply(ClosingExecu
tionResult.scala:65)",

"org.neo4j.cypher.internal.compatibility.ClosingExecutionResult$$anon$2$$anonfun$next$1.apply(ClosingExecu
tionResult.scala:64)",
            "org.neo4j.cypher.exceptionHandler$runSafely$.apply(exceptionHandler.scala:89)",

"org.neo4j.cypher.internal.compatibility.ClosingExecutionResult$$anon$2.next(ClosingExecutionResult.scala:
64)",

"org.neo4j.cypher.internal.compatibility.ClosingExecutionResult$$anon$2.next(ClosingExecutionResult.scala:
58)",
            "org.neo4j.cypher.internal.javacompat.ExecutionResult.next(ExecutionResult.java:236)",
            "org.neo4j.cypher.internal.javacompat.ExecutionResult.next(ExecutionResult.java:57)",

"org.neo4j.helpers.collection.ExceptionHandlingIterable$1.next(ExceptionHandlingIterable.java:72)",
            "org.neo4j.helpers.collection.IteratorWrapper.next(IteratorWrapper.java:49)",
            "org.neo4j.server.rest.repr.ListRepresentation.serialize(ListRepresentation.java:64)",
            "org.neo4j.server.rest.repr.Serializer.serialize(Serializer.java:78)",
            "org.neo4j.server.rest.repr.MappingSerializer.putList(MappingSerializer.java:66)",

"org.neo4j.server.rest.repr.CypherResultRepresentation.serialize(CypherResultRepresentation.java:58)",
            "org.neo4j.server.rest.repr.MappingRepresentation.serialize(MappingRepresentation.java:41)",
            "org.neo4j.server.rest.repr.OutputFormat.assemble(OutputFormat.java:232)",
            "org.neo4j.server.rest.repr.OutputFormat.formatRepresentation(OutputFormat.java:172)",
            "org.neo4j.server.rest.repr.OutputFormat.response(OutputFormat.java:155)",
            "org.neo4j.server.rest.repr.OutputFormat.ok(OutputFormat.java:70)",
            "org.neo4j.server.rest.web.CypherService.cypher(CypherService.java:140)",
            "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
            "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
            "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
            "java.lang.Thread.run(Thread.java:748)"
          ],
          "message": "/ by zero",
          "errors": [
            {
              "code": "Neo.DatabaseError.General.UnknownError",
              "stackTrace": "org.neo4j.cypher.internal.util.v3_4.ArithmeticException: / by zero\n\tat
org.neo4j.cypher.internal.runtime.interpreted.commands.expressions.Divide.apply(Divide.scala:38)\n\tat
org.neo4j.cypher.internal.runtime.interpreted.pipes.ProjectionPipe$$anonfun$internalCreateResults$1$$anonf
un$apply$1.apply(ProjectionPipe.scala:36)\n\tat
org.neo4j.cypher.internal.runtime.interpreted.pipes.ProjectionPipe$$anonfun$internalCreateResults$1$$anonf
un$apply$1.apply(ProjectionPipe.scala:34)\n\tat
scala.collection.immutable.Map$Map1.foreach(Map.scala:116)\n\tat
```

```
org.neo4j.cypher.internal.runtime.interpreted.pipes.ProjectionPipe$$anonfun$internalCreateResults$1.apply(
ProjectionPipe.scala:34)\n\tat
org.neo4j.cypher.internal.runtime.interpreted.pipes.ProjectionPipe$$anonfun$internalCreateResults$1.apply(
ProjectionPipe.scala:33)\n\tat scala.collection.Iterator$$anon$11.next(Iterator.scala:410)\n\tat
scala.collection.Iterator$$anon$11.next(Iterator.scala:410)\n\tat
org.neo4j.cypher.internal.compatibility.v3_4.runtime.ClosingIterator.next(ResultIterator.scala:74)\n\tat
org.neo4j.cypher.internal.compatibility.v3_4.runtime.ClosingIterator.next(ResultIterator.scala:48)\n\tat
org.neo4j.cypher.internal.compatibility.v3_4.runtime.PipeExecutionResult$$anon$2.next(PipeExecutionResult.
scala:72)\n\tat
org.neo4j.cypher.internal.compatibility.v3_4.runtime.PipeExecutionResult$$anon$2.next(PipeExecutionResult.
scala:70)\n\tat
org.neo4j.cypher.internal.compatibility.ClosingExecutionResult$$anon$2$$anonfun$next$1.apply(ClosingExecut
ionResult.scala:65)\n\tat
org.neo4j.cypher.internal.compatibility.ClosingExecutionResult$$anon$2$$anonfun$next$1.apply(ClosingExecut
ionResult.scala:64)\n\tat
org.neo4j.cypher.exceptionHandler$runSafely$.apply(exceptionHandler.scala:89)\n\tat
org.neo4j.cypher.internal.compatibility.ClosingExecutionResult$$anon$2.next(ClosingExecutionResult.scala:6
4)\n\tat
org.neo4j.cypher.internal.compatibility.ClosingExecutionResult$$anon$2.next(ClosingExecutionResult.scala:5
8)\n\tat org.neo4j.cypher.internal.javacompat.ExecutionResult.next(ExecutionResult.java:236)\n\tat
org.neo4j.cypher.internal.javacompat.ExecutionResult.next(ExecutionResult.java:57)\n\tat
org.neo4j.helpers.collection.ExceptionHandlingIterable$1.next(ExceptionHandlingIterable.java:72)\n\tat
org.neo4j.helpers.collection.IteratorWrapper.next(IteratorWrapper.java:49)\n\tat
org.neo4j.server.rest.repr.ListRepresentation.serialize(ListRepresentation.java:64)\n\tat
org.neo4j.server.rest.repr.Serializer.serialize(Serializer.java:78)\n\tat
org.neo4j.server.rest.repr.MappingSerializer.putList(MappingSerializer.java:66)\n\tat
org.neo4j.server.rest.repr.CypherResultRepresentation.serialize(CypherResultRepresentation.java:58)\n\tat
org.neo4j.server.rest.repr.MappingRepresentation.serialize(MappingRepresentation.java:41)\n\tat
org.neo4j.server.rest.repr.OutputFormat.assemble(OutputFormat.java:232)\n\tat
org.neo4j.server.rest.repr.OutputFormat.formatRepresentation(OutputFormat.java:172)\n\tat
org.neo4j.server.rest.repr.OutputFormat.response(OutputFormat.java:155)\n\tat
org.neo4j.server.rest.repr.OutputFormat.ok(OutputFormat.java:70)\n\tat
org.neo4j.server.rest.web.CypherService.cypher(CypherService.java:140)\n\tat
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)\n\tat
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)\n\tat
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)\n\tat
java.lang.reflect.Method.invoke(Method.java:498)\n\tat
com.sun.jersey.spi.container.JavaMethodInvokerFactory$1.invoke(JavaMethodInvokerFactory.java:60)\n\tat
com.sun.jersey.server.impl.model.method.dispatch.AbstractResourceMethodDispatchProvider$ResponseOutInvoker
._dispatch(AbstractResourceMethodDispatchProvider.java:205)\n\tat
com.sun.jersey.server.impl.model.method.dispatch.ResourceJavaMethodDispatcher.dispatch(ResourceJavaMethodD
ispatcher.java:75)\n\tat
org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatcher
.java:147)\n\tat com.sun.jersey.server.impl.uri.rules.HttpMethodRule.accept(HttpMethodRule.java:302)\n\tat
com.sun.jersey.server.impl.uri.rules.ResourceClassRule.accept(ResourceClassRule.java:108)\n\tat
com.sun.jersey.server.impl.uri.rules.RightHandPathRule.accept(RightHandPathRule.java:147)\n\tat
com.sun.jersey.server.impl.uri.rules.RootResourceClassesRule.accept(RootResourceClassesRule.java:84)\n\tat
com.sun.jersey.server.impl.application.WebApplicationImpl._handleRequest(WebApplicationImpl.java:1542)\n\t
at
com.sun.jersey.server.impl.application.WebApplicationImpl._handleRequest(WebApplicationImpl.java:1473)\n\t
at
com.sun.jersey.server.impl.application.WebApplicationImpl.handleRequest(WebApplicationImpl.java:1419)\n\ta
t
com.sun.jersey.server.impl.application.WebApplicationImpl.handleRequest(WebApplicationImpl.java:1409)\n\ta
t com.sun.jersey.spi.container.servlet.WebComponent.service(WebComponent.java:409)\n\tat
com.sun.jersey.spi.container.servlet.ServletContainer.service(ServletContainer.java:558)\n\tat
com.sun.jersey.spi.container.servlet.ServletContainer.service(ServletContainer.java:733)\n\tat
javax.servlet.http.HttpServlet.service(HttpServlet.java:790)\n\tat
org.eclipse.jetty.servlet.ServletHolder.handle(ServletHolder.java:860)\n\tat
org.eclipse.jetty.servlet.ServletHandler$CachedChain.doFilter(ServletHandler.java:1650)\n\tat
org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)\n\tat
org.eclipse.jetty.servlet.ServletHandler$CachedChain.doFilter(ServletHandler.java:1637)\n\tat
org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)\n\tat
org.eclipse.jetty.servlet.ServletHandler$CachedChain.doFilter(ServletHandler.java:1637)\n\tat
org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)\n\tat
org.eclipse.jetty.servlet.ServletHandler$CachedChain.doFilter(ServletHandler.java:1637)\n\tat
org.eclipse.jetty.servlet.ServletHandler.doHandle(ServletHandler.java:533)\n\tat
org.eclipse.jetty.server.handler.ScopedHandler.nextHandle(ScopedHandler.java:188)\n\tat
org.eclipse.jetty.server.session.SessionHandler.doHandle(SessionHandler.java:1595)\n\tat
org.eclipse.jetty.server.handler.ScopedHandler.nextHandle(ScopedHandler.java:188)\n\tat
org.eclipse.jetty.server.handler.ContextHandler.doHandle(ContextHandler.java:1253)\n\tat
org.eclipse.jetty.server.handler.ScopedHandler.nextScope(ScopedHandler.java:168)\n\tat
org.eclipse.jetty.servlet.ServletHandler.doScope(ServletHandler.java:473)\n\tat
org.eclipse.jetty.server.session.SessionHandler.doScope(SessionHandler.java:1564)\n\tat
org.eclipse.jetty.server.handler.ScopedHandler.nextScope(ScopedHandler.java:166)\n\tat
org.eclipse.jetty.server.handler.ContextHandler.doScope(ContextHandler.java:1155)\n\tat
org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:141)\n\tat
org.eclipse.jetty.server.handler.HandlerList.handle(HandlerList.java:61)\n\tat
org.eclipse.jetty.server.handler.HandlerWrapper.handle(HandlerWrapper.java:132)\n\tat
org.eclipse.jetty.server.Server.handle(Server.java:530)\n\tat
org.eclipse.jetty.server.HttpChannel.handle(HttpChannel.java:347)\n\tat
org.eclipse.jetty.server.HttpConnection.onFillable(HttpConnection.java:256)\n\tat
org.eclipse.jetty.io.AbstractConnection$ReadCallback.succeeded(AbstractConnection.java:279)\n\tat
org.eclipse.jetty.io.FillInterest.fillable(FillInterest.java:102)\n\tat
org.eclipse.jetty.io.ChannelEndPoint$2.run(ChannelEndPoint.java:124)\n\tat
org.eclipse.jetty.util.thread.strategy.EatWhatYouKill.doProduce(EatWhatYouKill.java:247)\n\tat
org.eclipse.jetty.util.thread.strategy.EatWhatYouKill.produce(EatWhatYouKill.java:140)\n\tat
org.eclipse.jetty.util.thread.strategy.EatWhatYouKill.run(EatWhatYouKill.java:131)\n\tat
org.eclipse.jetty.util.thread.ReservedThreadExecutor$ReservedThread.run(ReservedThreadExecutor.java:382)\n
\tat org.eclipse.jetty.util.thread.QueuedThreadPool.runJob(QueuedThreadPool.java:708)\n\tat
org.eclipse.jetty.util.thread.QueuedThreadPool$2.run(QueuedThreadPool.java:626)\n\tat
java.lang.Thread.run(Thread.java:748)\n",
              "message": "/ by zero"
            }
          ]
        },
        "fullname": "org.neo4j.cypher.ArithmeticException",
        "stackTrace": [
          "org.neo4j.cypher.exceptionHandler$.arithmeticException(exceptionHandler.scala:29)",
          "org.neo4j.cypher.exceptionHandler$.arithmeticException(exceptionHandler.scala:26)",

"org.neo4j.cypher.internal.util.v3_4.ArithmeticException.mapToPublic(CypherException.scala:125)",
          "org.neo4j.cypher.exceptionHandler$runSafely$.apply(exceptionHandler.scala:94)",

"org.neo4j.cypher.internal.compatibility.ClosingExecutionResult$$anon$2.next(ClosingExecutionResult.scala:
64)",

"org.neo4j.cypher.internal.compatibility.ClosingExecutionResult$$anon$2.next(ClosingExecutionResult.scala:
58)",
          "org.neo4j.cypher.internal.javacompat.ExecutionResult.next(ExecutionResult.java:236)",
          "org.neo4j.cypher.internal.javacompat.ExecutionResult.next(ExecutionResult.java:57)",

"org.neo4j.helpers.collection.ExceptionHandlingIterable$1.next(ExceptionHandlingIterable.java:72)",
          "org.neo4j.helpers.collection.IteratorWrapper.next(IteratorWrapper.java:49)",
          "org.neo4j.server.rest.repr.ListRepresentation.serialize(ListRepresentation.java:64)",
          "org.neo4j.server.rest.repr.Serializer.serialize(Serializer.java:78)",
          "org.neo4j.server.rest.repr.MappingSerializer.putList(MappingSerializer.java:66)",

"org.neo4j.server.rest.repr.CypherResultRepresentation.serialize(CypherResultRepresentation.java:58)",
          "org.neo4j.server.rest.repr.MappingRepresentation.serialize(MappingRepresentation.java:41)",
          "org.neo4j.server.rest.repr.OutputFormat.assemble(OutputFormat.java:232)",
          "org.neo4j.server.rest.repr.OutputFormat.formatRepresentation(OutputFormat.java:172)",
          "org.neo4j.server.rest.repr.OutputFormat.response(OutputFormat.java:155)",
          "org.neo4j.server.rest.repr.OutputFormat.ok(OutputFormat.java:70)",
          "org.neo4j.server.rest.web.CypherService.cypher(CypherService.java:140)",
          "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
          "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
          "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
          "java.lang.Thread.run(Thread.java:748)"
        ],
        "message": "/ by zero",
        "errors": [
          {
            "code": "Neo.ClientError.Statement.ArithmeticError",
            "message": "/ by zero"
          }
        ]
      },
      "fullname": "org.neo4j.kernel.impl.query.QueryExecutionKernelException",
      "stackTrace": [
        "org.neo4j.cypher.internal.javacompat.ExecutionResult.converted(ExecutionResult.java:405)",
        "org.neo4j.cypher.internal.javacompat.ExecutionResult.next(ExecutionResult.java:240)",
        "org.neo4j.cypher.internal.javacompat.ExecutionResult.next(ExecutionResult.java:57)",

"org.neo4j.helpers.collection.ExceptionHandlingIterable$1.next(ExceptionHandlingIterable.java:72)",
        "org.neo4j.helpers.collection.IteratorWrapper.next(IteratorWrapper.java:49)",
        "org.neo4j.server.rest.repr.ListRepresentation.serialize(ListRepresentation.java:64)",
        "org.neo4j.server.rest.repr.Serializer.serialize(Serializer.java:78)",
        "org.neo4j.server.rest.repr.MappingSerializer.putList(MappingSerializer.java:66)",

"org.neo4j.server.rest.repr.CypherResultRepresentation.serialize(CypherResultRepresentation.java:58)",
        "org.neo4j.server.rest.repr.MappingRepresentation.serialize(MappingRepresentation.java:41)",
        "org.neo4j.server.rest.repr.OutputFormat.assemble(OutputFormat.java:232)",
        "org.neo4j.server.rest.repr.OutputFormat.formatRepresentation(OutputFormat.java:172)",
        "org.neo4j.server.rest.repr.OutputFormat.response(OutputFormat.java:155)",
        "org.neo4j.server.rest.repr.OutputFormat.ok(OutputFormat.java:70)",
        "org.neo4j.server.rest.web.CypherService.cypher(CypherService.java:140)",
        "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
        "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
        "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
        "java.lang.Thread.run(Thread.java:748)"
      ],
      "message": "/ by zero",
      "errors": [
        {
          "code": "Neo.ClientError.Statement.ArithmeticError",
          "message": "/ by zero"
        }
      ]
    },
    "fullname": "org.neo4j.graphdb.QueryExecutionException",
    "stackTrace": [

"org.neo4j.kernel.impl.query.QueryExecutionKernelException.asUserException(QueryExecutionKernelException.j
ava:35)",
      "org.neo4j.cypher.internal.javacompat.ExecutionResult.converted(ExecutionResult.java:405)",
      "org.neo4j.cypher.internal.javacompat.ExecutionResult.next(ExecutionResult.java:240)",
      "org.neo4j.cypher.internal.javacompat.ExecutionResult.next(ExecutionResult.java:57)",
      "org.neo4j.helpers.collection.ExceptionHandlingIterable$1.next(ExceptionHandlingIterable.java:72)",
      "org.neo4j.helpers.collection.IteratorWrapper.next(IteratorWrapper.java:49)",
      "org.neo4j.server.rest.repr.ListRepresentation.serialize(ListRepresentation.java:64)",
      "org.neo4j.server.rest.repr.Serializer.serialize(Serializer.java:78)",
      "org.neo4j.server.rest.repr.MappingSerializer.putList(MappingSerializer.java:66)",

"org.neo4j.server.rest.repr.CypherResultRepresentation.serialize(CypherResultRepresentation.java:58)",
      "org.neo4j.server.rest.repr.MappingRepresentation.serialize(MappingRepresentation.java:41)",
      "org.neo4j.server.rest.repr.OutputFormat.assemble(OutputFormat.java:232)",
      "org.neo4j.server.rest.repr.OutputFormat.formatRepresentation(OutputFormat.java:172)",
      "org.neo4j.server.rest.repr.OutputFormat.response(OutputFormat.java:155)",
      "org.neo4j.server.rest.repr.OutputFormat.ok(OutputFormat.java:70)",
      "org.neo4j.server.rest.web.CypherService.cypher(CypherService.java:140)",
      "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
      "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
      "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
      "java.lang.Thread.run(Thread.java:748)"
    ],
    "message": "/ by zero",
    "errors": [
      {
        "code": "Neo.ClientError.Statement.ArithmeticError",
        "message": "/ by zero"
      }
    ]
  },
  "errors": [
    {
      "code": "Neo.ClientError.Request.InvalidFormat",
      "message": "/ by zero"
    }
  ]
}
```

Property values

The REST API allows setting properties on nodes and relationships through direct RESTful operations. However, there are restrictions as to what types of values can be used as property values. Allowed value types are as follows:

- Numbers: Both integer values, with capacity as Java's Long type, and floating points, with capacity as Java's Double.
- Booleans.
- Strings.
- Arrays of the basic types above.

Arrays

There are two important points to be made about array values. First, all values in the array must be of the same type. That means either all integers, all floats, all booleans or all strings. Mixing types is not currently supported.

Second, storing empty arrays is only possible given certain preconditions. Because the JSON transfer format does not contain type information for arrays, type is inferred from the values in the array. If the array is empty, the Neo4j Server cannot determine the type. In these cases, it will check if an array is already stored for the given property, and will use the stored array's type when storing the empty array. If no array exists already, the server will reject the request.

Property keys

You can list all property keys ever used in the database. This includes and property keys you have used, but deleted.

There is currently no way to tell which ones are in use and which ones are not, short of walking the entire set of properties in the database.

List all property keys

Example request

- **GET** http://localhost:7474/db/data/propertykeys
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ "cost", "name", "foo" ]
```

Nodes

Create node

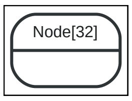


Figure 12. Final Graph

Example request

- **POST** http://localhost:7474/db/data/node
- **Accept:** application/json; charset=UTF-8

Example response

- **201:** Created
- **Content-Type:** application/json;charset=utf-8
- **Location:** http://localhost:7474/db/data/node/32

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 32,
    "labels" : [ ]
  },
  "paged_traverse" :
"http://localhost:7474/db/data/node/32/paged/traverse/{returnType}{?pageSize,leaseTime}",
  "outgoing_relationships" : "http://localhost:7474/db/data/node/32/relationships/out",
  "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/32/relationships/out/{-
list|&|types}",
  "create_relationship" : "http://localhost:7474/db/data/node/32/relationships",
  "labels" : "http://localhost:7474/db/data/node/32/labels",
  "traverse" : "http://localhost:7474/db/data/node/32/traverse/{returnType}",
  "all_relationships" : "http://localhost:7474/db/data/node/32/relationships/all",
  "all_typed_relationships" : "http://localhost:7474/db/data/node/32/relationships/all/{-list|&|types}",
  "property" : "http://localhost:7474/db/data/node/32/properties/{key}",
  "self" : "http://localhost:7474/db/data/node/32",
  "incoming_relationships" : "http://localhost:7474/db/data/node/32/relationships/in",
  "properties" : "http://localhost:7474/db/data/node/32/properties",
  "incoming_typed_relationships" : "http://localhost:7474/db/data/node/32/relationships/in/{-
list|&|types}",
  "data" : { }
}
```

Create node with properties

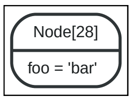


Figure 13. Final Graph

Example request

- **POST** http://localhost:7474/db/data/node
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "foo" : "bar"
}
```

Example response

- **201:** Created
- **Content-Length:** 1223
- **Content-Type:** application/json;charset=utf-8
- **Location:** http://localhost:7474/db/data/node/28

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 28,
    "labels" : [ ]
  },
  "paged_traverse" :
"http://localhost:7474/db/data/node/28/paged/traverse/{returnType}{?pageSize,leaseTime}",
  "outgoing_relationships" : "http://localhost:7474/db/data/node/28/relationships/out",
  "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/28/relationships/out/{-
list|&|types}",
  "create_relationship" : "http://localhost:7474/db/data/node/28/relationships",
  "labels" : "http://localhost:7474/db/data/node/28/labels",
  "traverse" : "http://localhost:7474/db/data/node/28/traverse/{returnType}",
  "all_relationships" : "http://localhost:7474/db/data/node/28/relationships/all",
  "all_typed_relationships" : "http://localhost:7474/db/data/node/28/relationships/all/{-list|&|types}",
  "property" : "http://localhost:7474/db/data/node/28/properties/{key}",
  "self" : "http://localhost:7474/db/data/node/28",
  "incoming_relationships" : "http://localhost:7474/db/data/node/28/relationships/in",
  "properties" : "http://localhost:7474/db/data/node/28/properties",
  "incoming_typed_relationships" : "http://localhost:7474/db/data/node/28/relationships/in/{-
list|&|types}",
  "data" : {
    "foo" : "bar"
  }
}
```

Get node

Note that the response contains URI/templates for the available operations for getting properties and relationships.

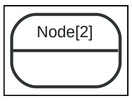


Figure 14. Final Graph

Example request

- **GET** http://localhost:7474/db/data/node/2
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 2,
    "labels" : [ ]
  },
  "paged_traverse" :
"http://localhost:7474/db/data/node/2/paged/traverse/{returnType}{?pageSize,leaseTime}",
  "outgoing_relationships" : "http://localhost:7474/db/data/node/2/relationships/out",
  "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/2/relationships/out/{-
list|&|types}",
  "create_relationship" : "http://localhost:7474/db/data/node/2/relationships",
  "labels" : "http://localhost:7474/db/data/node/2/labels",
  "traverse" : "http://localhost:7474/db/data/node/2/traverse/{returnType}",
  "all_relationships" : "http://localhost:7474/db/data/node/2/relationships/all",
  "all_typed_relationships" : "http://localhost:7474/db/data/node/2/relationships/all/{-list|&|types}",
  "property" : "http://localhost:7474/db/data/node/2/properties/{key}",
  "self" : "http://localhost:7474/db/data/node/2",
  "incoming_relationships" : "http://localhost:7474/db/data/node/2/relationships/in",
  "properties" : "http://localhost:7474/db/data/node/2/properties",
  "incoming_typed_relationships" : "http://localhost:7474/db/data/node/2/relationships/in/{-
list|&|types}",
  "data" : { }
}
```

Get non-existent node

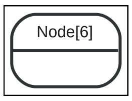


Figure 15. Final Graph

Example request

- **GET** http://localhost:7474/db/data/node/600000
- **Accept:** application/json; charset=UTF-8

Example response

- **404:** Not Found
- **Content-Type:** application/json;charset=utf-8

```
{
  "message": "Cannot find node with id [600000] in database.",
  "exception": "NodeNotFoundException",
  "fullname": "org.neo4j.server.rest.web.NodeNotFoundException",
  "stackTrace": [
    "org.neo4j.server.rest.web.DatabaseActions.node(DatabaseActions.java:168)",
    "org.neo4j.server.rest.web.DatabaseActions.getNode(DatabaseActions.java:213)",
    "org.neo4j.server.rest.web.RestfulGraphDatabase.getNode(RestfulGraphDatabase.java:272)",
    "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
    "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
    "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
    "java.lang.Thread.run(Thread.java:745)"
  ],
  "cause": {
    "exception": "NotFoundException",
    "cause": {
      "exception": "EntityNotFoundException",
      "fullname": "org.neo4j.internal.kernel.api.exceptions.EntityNotFoundException",
      "stackTrace": [
        "org.neo4j.kernel.impl.factory.GraphDatabaseFacade.getNodeById(GraphDatabaseFacade.java:313)",
        "org.neo4j.server.rest.web.DatabaseActions.node(DatabaseActions.java:164)",
        "org.neo4j.server.rest.web.DatabaseActions.getNode(DatabaseActions.java:213)",
        "org.neo4j.server.rest.web.RestfulGraphDatabase.getNode(RestfulGraphDatabase.java:272)",
        "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
        "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
        "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
        "java.lang.Thread.run(Thread.java:745)"
      ],
      "message": "Unable to load NODE with id 600000.",
      "errors": [
        {
          "code": "Neo.ClientError.Statement.EntityNotFound",
          "message": "Unable to load NODE with id 600000."
        }
      ]
    },
    "fullname": "org.neo4j.graphdb.NotFoundException",
    "stackTrace": [
      "org.neo4j.kernel.impl.factory.GraphDatabaseFacade.getNodeById(GraphDatabaseFacade.java:313)",
      "org.neo4j.server.rest.web.DatabaseActions.node(DatabaseActions.java:164)",
      "org.neo4j.server.rest.web.DatabaseActions.getNode(DatabaseActions.java:213)",
      "org.neo4j.server.rest.web.RestfulGraphDatabase.getNode(RestfulGraphDatabase.java:272)",
      "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
      "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
      "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
      "java.lang.Thread.run(Thread.java:745)"
    ],
    "message": "Node 600000 not found",
    "errors": [
      {
        "code": "Neo.ClientError.Statement.EntityNotFound",
        "message": "Node 600000 not found"
      }
    ]
  },
  "errors": [
    {
      "code": "Neo.ClientError.Statement.EntityNotFound",
      "message": "Cannot find node with id [600000] in database."
    }
  ]
}
```

Delete node

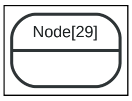


Figure 16. Starting Graph

Figure 17. Final Graph

Example request

- **DELETE** http://localhost:7474/db/data/node/29
- **Accept:** application/json; charset=UTF-8

Example response

- **204:** No Content

```

```

Nodes with relationships cannot be deleted

The relationships on a node has to be deleted before the node can be deleted.

You can use `DETACH DELETE` in Cypher to delete nodes and their relationships in one go.

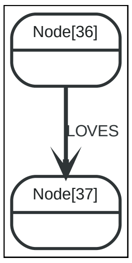


Figure 18. Starting Graph

Example request

- **DELETE** http://localhost:7474/db/data/node/36
- **Accept:** application/json; charset=UTF-8

Example response

- **409:** Conflict
- **Content-Type:** application/json;charset=utf-8

```
{
  "message": "The node with id 36 cannot be deleted. Check that the node is orphaned before deletion.",
  "exception": "ConstraintViolationException",
  "fullname": "org.neo4j.graphdb.ConstraintViolationException",
  "stackTrace": [
    "org.neo4j.server.rest.web.DatabaseActions.deleteNode(DatabaseActions.java:223)",
    "org.neo4j.server.rest.web.RestfulGraphDatabase.deleteNode(RestfulGraphDatabase.java:286)",
    "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
    "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
    "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
    "java.lang.Thread.run(Thread.java:745)"
  ],
  "errors": [
    {
      "code": "Neo.ClientError.Schema.ConstraintValidationFailed",
      "message": "The node with id 36 cannot be deleted. Check that the node is orphaned before deletion."
    }
  ]
}
```

Relationships

Relationships are a first class citizen in the Neo4j REST API. They can be accessed either stand-alone or through the nodes they are attached to.

The general pattern to get relationships from a node is:

```
GET http://localhost:7474/db/data/node/123/relationships/{dir}/{-list|&|types}
```

Where `dir` is one of `all`, `in`, `out` and `types` is an ampersand-separated list of types. See the examples below for more information.

Get Relationship by ID

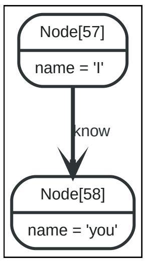


Figure 19. Final Graph

Example request

- **GET** http://localhost:7474/db/data/relationship/10
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 10,
    "type" : "know"
  },
  "property" : "http://localhost:7474/db/data/relationship/10/properties/{key}",
  "start" : "http://localhost:7474/db/data/node/57",
  "self" : "http://localhost:7474/db/data/relationship/10",
  "end" : "http://localhost:7474/db/data/node/58",
  "type" : "know",
  "properties" : "http://localhost:7474/db/data/relationship/10/properties",
  "data" : { }
}
```

Create relationship

Upon successful creation of a relationship, the new relationship is returned.

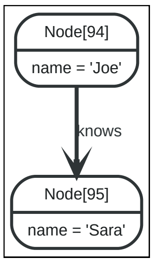


Figure 20. Starting Graph

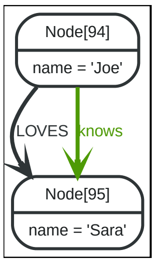


Figure 21. Final Graph

Example request

- **POST** http://localhost:7474/db/data/node/94/relationships
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "to" : "http://localhost:7474/db/data/node/95",
  "type" : "LOVES"
}
```

Example response

- **201:** Created
- **Content-Type:** application/json;charset=utf-8
- **Location:** http://localhost:7474/db/data/relationship/57

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 57,
    "type" : "LOVES"
  },
  "property" : "http://localhost:7474/db/data/relationship/57/properties/{key}",
  "start" : "http://localhost:7474/db/data/node/94",
  "self" : "http://localhost:7474/db/data/relationship/57",
  "end" : "http://localhost:7474/db/data/node/95",
  "type" : "LOVES",
  "properties" : "http://localhost:7474/db/data/relationship/57/properties",
  "data" : { }
}
```

Create a relationship with properties

Upon successful creation of a relationship, the new relationship is returned.

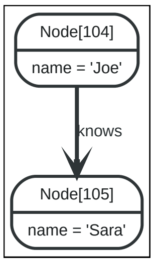


Figure 22. Starting Graph

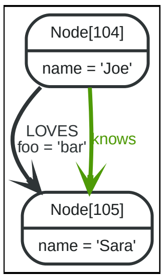


Figure 23. Final Graph

Example request

- **POST** http://localhost:7474/db/data/node/104/relationships
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "to" : "http://localhost:7474/db/data/node/105",
  "type" : "LOVES",
  "data" : {
    "foo" : "bar"
  }
}
```

Example response

- **201:** Created
- **Content-Type:** application/json;charset=utf-8
- **Location:** http://localhost:7474/db/data/relationship/64

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 64,
    "type" : "LOVES"
  },
  "property" : "http://localhost:7474/db/data/relationship/64/properties/{key}",
  "start" : "http://localhost:7474/db/data/node/104",
  "self" : "http://localhost:7474/db/data/relationship/64",
  "end" : "http://localhost:7474/db/data/node/105",
  "type" : "LOVES",
  "properties" : "http://localhost:7474/db/data/relationship/64/properties",
  "data" : {
    "foo" : "bar"
  }
}
```

Delete relationship

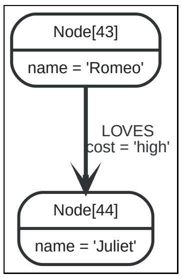


Figure 24. Starting Graph

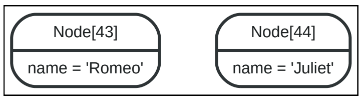


Figure 25. Final Graph

Example request

- **DELETE** http://localhost:7474/db/data/relationship/3
- **Accept:** application/json; charset=UTF-8

Example response

- **204:** No Content

```

```

Get all properties on a relationship

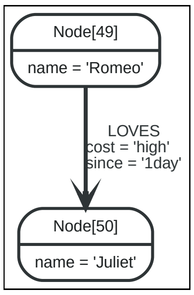


Figure 26. Final Graph

Example request

- **GET** http://localhost:7474/db/data/relationship/6/properties
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "cost" : "high",
  "since" : "1day"
}
```

Set all properties on a relationship

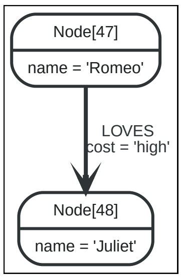


Figure 27. Starting Graph

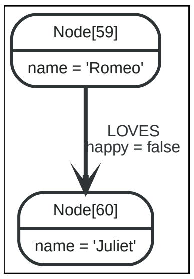


Figure 28. Final Graph

Example request

- **PUT** http://localhost:7474/db/data/relationship/11/properties
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "happy" : false
}
```

Example response

- **204:** No Content

```

```

Get single property on a relationship

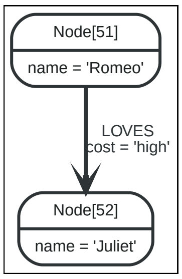


Figure 29. Final Graph

Example request

- **GET** http://localhost:7474/db/data/relationship/7/properties/cost
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
"high"
```

Set single property on a relationship

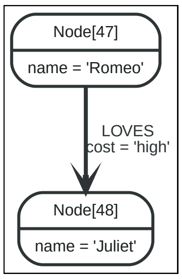


Figure 30. Starting Graph

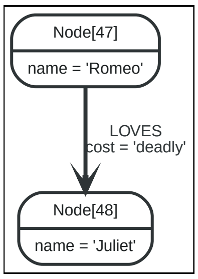


Figure 31. Final Graph

Example request

- **PUT** http://localhost:7474/db/data/relationship/5/properties/cost
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
"deadly"
```

Example response

- **204:** No Content

```

```

Get all relationships

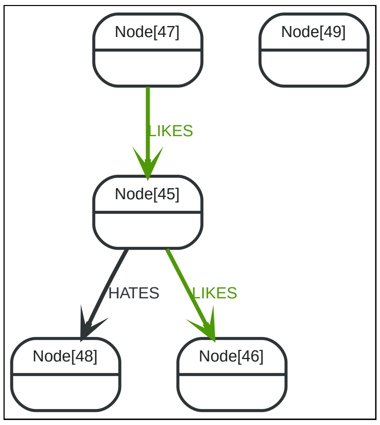


Figure 32. Final Graph

Example request

- **GET** http://localhost:7474/db/data/node/45/relationships/all
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "extensions" : { },
  "metadata" : {
    "id" : 29,
    "type" : "HATES"
  },
  "data" : { },
  "property" : "http://localhost:7474/db/data/relationship/29/properties/{key}",
  "start" : "http://localhost:7474/db/data/node/45",
  "self" : "http://localhost:7474/db/data/relationship/29",
  "end" : "http://localhost:7474/db/data/node/48",
  "type" : "HATES",
  "properties" : "http://localhost:7474/db/data/relationship/29/properties"
}, {
  "extensions" : { },
  "metadata" : {
    "id" : 28,
    "type" : "LIKES"
  },
  "data" : { },
  "property" : "http://localhost:7474/db/data/relationship/28/properties/{key}",
  "start" : "http://localhost:7474/db/data/node/47",
  "self" : "http://localhost:7474/db/data/relationship/28",
  "end" : "http://localhost:7474/db/data/node/45",
  "type" : "LIKES",
  "properties" : "http://localhost:7474/db/data/relationship/28/properties"
}, {
  "extensions" : { },
  "metadata" : {
    "id" : 27,
    "type" : "LIKES"
  },
  "data" : { },
  "property" : "http://localhost:7474/db/data/relationship/27/properties/{key}",
  "start" : "http://localhost:7474/db/data/node/45",
  "self" : "http://localhost:7474/db/data/relationship/27",
  "end" : "http://localhost:7474/db/data/node/46",
  "type" : "LIKES",
  "properties" : "http://localhost:7474/db/data/relationship/27/properties"
} ]
```

Get incoming relationships

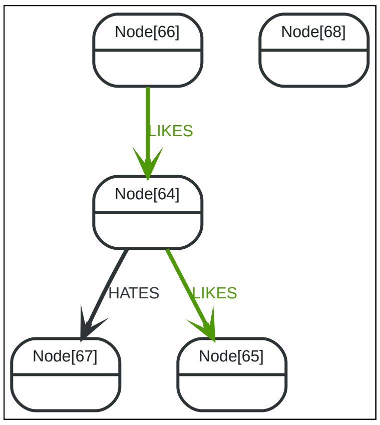


Figure 33. Final Graph

Example request

- **GET** http://localhost:7474/db/data/node/64/relationships/in
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "extensions" : { },
  "metadata" : {
    "id" : 39,
    "type" : "LIKES"
  },
  "data" : { },
  "property" : "http://localhost:7474/db/data/relationship/39/properties/{key}",
  "start" : "http://localhost:7474/db/data/node/66",
  "self" : "http://localhost:7474/db/data/relationship/39",
  "end" : "http://localhost:7474/db/data/node/64",
  "type" : "LIKES",
  "properties" : "http://localhost:7474/db/data/relationship/39/properties"
} ]
```

Get outgoing relationships

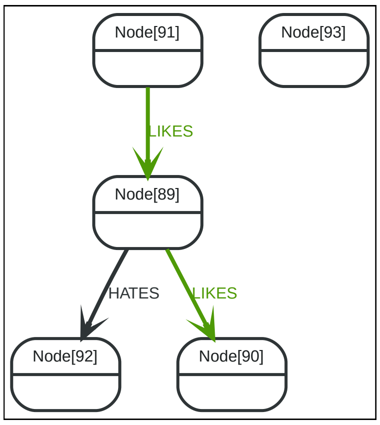


Figure 34. Final Graph

Example request

- **GET** http://localhost:7474/db/data/node/89/relationships/out
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "extensions" : { },
  "metadata" : {
    "id" : 55,
    "type" : "HATES"
  },
  "data" : { },
  "property" : "http://localhost:7474/db/data/relationship/55/properties/{key}",
  "start" : "http://localhost:7474/db/data/node/89",
  "self" : "http://localhost:7474/db/data/relationship/55",
  "end" : "http://localhost:7474/db/data/node/92",
  "type" : "HATES",
  "properties" : "http://localhost:7474/db/data/relationship/55/properties"
}, {
  "extensions" : { },
  "metadata" : {
    "id" : 53,
    "type" : "LIKES"
  },
  "data" : { },
  "property" : "http://localhost:7474/db/data/relationship/53/properties/{key}",
  "start" : "http://localhost:7474/db/data/node/89",
  "self" : "http://localhost:7474/db/data/relationship/53",
  "end" : "http://localhost:7474/db/data/node/90",
  "type" : "LIKES",
  "properties" : "http://localhost:7474/db/data/relationship/53/properties"
} ]
```

Get typed relationships

Note that the "&" needs to be encoded like "%26" for example when using cURL *(http://curl.haxx.se/)* from the terminal.

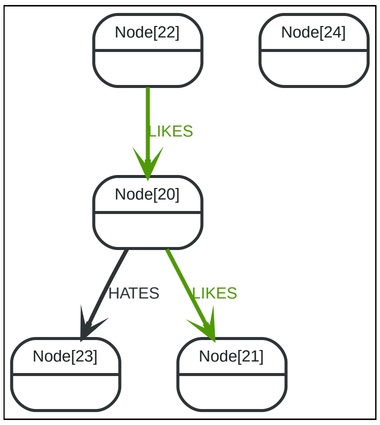


Figure 35. Final Graph

Example request

- **GET** http://localhost:7474/db/data/node/20/relationships/all/LIKES&HATES
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "extensions" : { },
  "metadata" : {
    "id" : 14,
    "type" : "HATES"
  },
  "data" : { },
  "property" : "http://localhost:7474/db/data/relationship/14/properties/{key}",
  "start" : "http://localhost:7474/db/data/node/20",
  "self" : "http://localhost:7474/db/data/relationship/14",
  "end" : "http://localhost:7474/db/data/node/23",
  "type" : "HATES",
  "properties" : "http://localhost:7474/db/data/relationship/14/properties"
}, {
  "extensions" : { },
  "metadata" : {
    "id" : 13,
    "type" : "LIKES"
  },
  "data" : { },
  "property" : "http://localhost:7474/db/data/relationship/13/properties/{key}",
  "start" : "http://localhost:7474/db/data/node/22",
  "self" : "http://localhost:7474/db/data/relationship/13",
  "end" : "http://localhost:7474/db/data/node/20",
  "type" : "LIKES",
  "properties" : "http://localhost:7474/db/data/relationship/13/properties"
}, {
  "extensions" : { },
  "metadata" : {
    "id" : 12,
    "type" : "LIKES"
  },
  "data" : { },
  "property" : "http://localhost:7474/db/data/relationship/12/properties/{key}",
  "start" : "http://localhost:7474/db/data/node/20",
  "self" : "http://localhost:7474/db/data/relationship/12",
  "end" : "http://localhost:7474/db/data/node/21",
  "type" : "LIKES",
  "properties" : "http://localhost:7474/db/data/relationship/12/properties"
} ]
```

Get relationships on a node without relationships

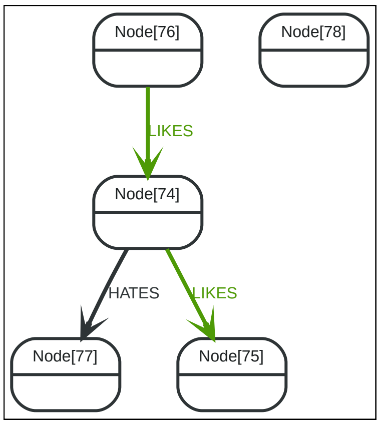


Figure 36. Final Graph

Example request

- **GET** http://localhost:7474/db/data/node/78/relationships/all
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ ]
```

Relationship types

Get relationship types

Example request

- **GET** http://localhost:7474/db/data/relationship/types
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json

```
[ "KNOWS", "LOVES" ]
```

Node properties

Set property on node

Setting different properties will retain the existing ones for this node. Note that a single value are submitted not as a map but just as a value (which is valid JSON) like in the example below.

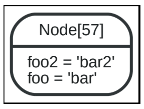


Figure 37. Final Graph

Example request

- **PUT** http://localhost:7474/db/data/node/57/properties/foo
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
"bar"
```

Example response

- **204:** No Content

```

```

Update node properties

This will replace all existing properties on the node with the new set of attributes.

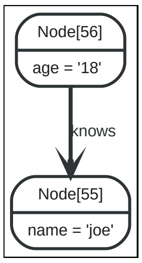


Figure 38. Final Graph

Example request

- **PUT** http://localhost:7474/db/data/node/56/properties
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "age" : "18"
}
```

Example response

- **204:** No Content

```
```

Get properties for node

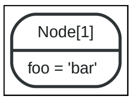


Figure 39. Final Graph

Example request

- **GET** http://localhost:7474/db/data/node/1/properties
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "foo" : "bar"
}
```

Get property for node

Get a single node property from a node.

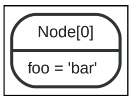


Figure 40. Final Graph

Example request

- **GET** http://localhost:7474/db/data/node/0/properties/foo
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
"bar"
```

Property values can not be null

This example shows the response you get when trying to set a property to null.

Example request

- **POST** http://localhost:7474/db/data/node
- **Accept:** application/json; charset=UTF-8

* **Content-Type:** application/json

```
{
  "foo" : null
}
```

Example response

* **400:** Bad Request
* **Content-Type:** application/json;charset=utf-8

```
{
  "message": "Could not set property \"foo\", unsupported type: null",
  "exception": "PropertyValueException",
  "fullname": "org.neo4j.server.rest.web.PropertyValueException",
  "stackTrace": [
    "org.neo4j.server.rest.domain.PropertySettingStrategy.setProperty(PropertySettingStrategy.java:134)",
    "org.neo4j.server.rest.domain.PropertySettingStrategy.setProperties(PropertySettingStrategy.java:85)",
    "org.neo4j.server.rest.web.DatabaseActions.createNode(DatabaseActions.java:199)",
    "org.neo4j.server.rest.web.RestfulGraphDatabase.createNode(RestfulGraphDatabase.java:249)",
    "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
    "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
    "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
    "java.lang.Thread.run(Thread.java:745)"
  ],
  "errors": [
    {
      "code": "Neo.ClientError.Statement.ArgumentError",
      "message": "Could not set property \"foo\", unsupported type: null"
    }
  ]
}
```

Property values can not be nested

Nesting properties is not supported. You could for example store the nested JSON as a string instead.

Example request

* **POST** http://localhost:7474/db/data/node/
* **Accept:** application/json; charset=UTF-8
* **Content-Type:** application/json

```
{
  "foo" : {
    "bar" : "baz"
  }
}
```

Example response

* **400:** Bad Request
* **Content-Type:** application/json;charset=utf-8

```
{
  "message": "Could not set property \"foo\", unsupported type: {bar\u003dbaz}",
  "exception": "PropertyValueException",
  "fullname": "org.neo4j.server.rest.web.PropertyValueException",
  "stackTrace": [
    "org.neo4j.server.rest.domain.PropertySettingStrategy.setProperty(PropertySettingStrategy.java:134)",
    "org.neo4j.server.rest.domain.PropertySettingStrategy.setProperties(PropertySettingStrategy.java:85)",
    "org.neo4j.server.rest.web.DatabaseActions.createNode(DatabaseActions.java:199)",
    "org.neo4j.server.rest.web.RestfulGraphDatabase.createNode(RestfulGraphDatabase.java:249)",
    "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
    "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
    "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
    "java.lang.Thread.run(Thread.java:745)"
  ],
  "errors": [
    {
      "code": "Neo.ClientError.Statement.ArgumentError",
      "message": "Could not set property \"foo\", unsupported type: {bar\u003dbaz}"
    }
  ]
}
```

Delete all properties from node

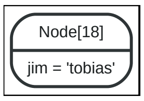


Figure 41. Starting Graph

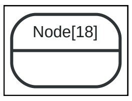


Figure 42. Final Graph

Example request

- **DELETE** http://localhost:7474/db/data/node/18/properties
- **Accept:** application/json; charset=UTF-8

Example response

- **204:** No Content

```

```

Delete a named property from a node

To delete a single property from a node, see the example below.

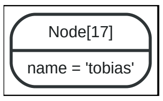


Figure 43. Starting Graph

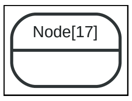


Figure 44. Final Graph

Example request

- **DELETE** http://localhost:7474/db/data/node/17/properties/name
- **Accept:** application/json; charset=UTF-8

Example response

- **204:** No Content

```

```

Relationship properties

Update relationship properties

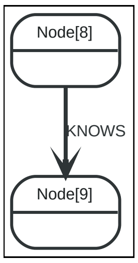


Figure 45. Starting Graph

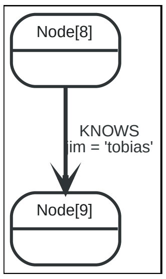


Figure 46. Final Graph

Example request

- **PUT** http://localhost:7474/db/data/relationship/4/properties
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "jim" : "tobias"
}
```

Example response

- **204:** No Content

```

```

Remove properties from a relationship

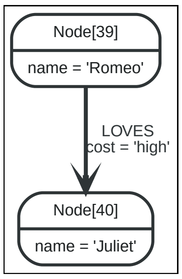


Figure 47. Starting Graph

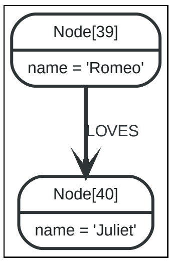


Figure 48. Final Graph

Example request

- **DELETE** http://localhost:7474/db/data/relationship/1/properties
- **Accept:** application/json; charset=UTF-8

Example response

- **204:** No Content

```

```

Remove property from a relationship

See the example request below.

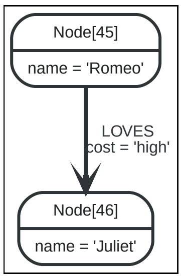


Figure 49. Starting Graph

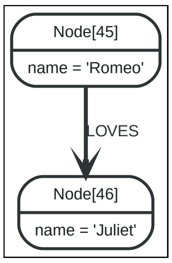


Figure 50. Final Graph

Example request

- **DELETE** http://localhost:7474/db/data/relationship/4/properties/cost
- **Accept:** application/json; charset=UTF-8

Example response

- **204:** No Content

```

```

Remove non-existent property from a relationship

Attempting to remove a property that doesn't exist results in an error.

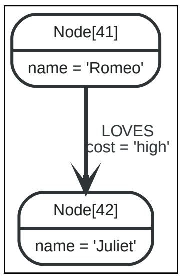


Figure 51. Starting Graph

Example request

- **DELETE** http://localhost:7474/db/data/relationship/2/properties/non-existent
- **Accept:** application/json; charset=UTF-8

Example response

- **404:** Not Found
- **Content-Type:** application/json;charset=utf-8

```
{
  "message": "(41)-[LOVES,2]-\u003e(42) does not have a property \"non-existent\"",
  "exception": "NoSuchPropertyException",
  "fullname": "org.neo4j.server.rest.web.NoSuchPropertyException",
  "stackTrace": [
    "org.neo4j.server.rest.web.DatabaseActions.removeRelationshipProperty(DatabaseActions.java:663)",

"org.neo4j.server.rest.web.RestfulGraphDatabase.deleteRelationshipProperty(RestfulGraphDatabase.java:781)"
,
    "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
    "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
    "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
    "java.lang.Thread.run(Thread.java:745)"
  ],
  "errors": [
    {
      "code": "Neo.ClientError.Statement.PropertyNotFound",
      "message": "(41)-[LOVES,2]-\u003e(42) does not have a property \"non-existent\""
    }
  ]
}
```

Remove properties from a non-existing relationship

Attempting to remove all properties from a relationship which doesn't exist results in an error.

Example request

- **DELETE** http://localhost:7474/db/data/relationship/1234/properties
- **Accept:** application/json; charset=UTF-8

Example response

* **404:** Not Found
* **Content-Type:** application/json;charset=utf-8

```
{
  "message": "org.neo4j.graphdb.NotFoundException: Relationship 1234 not found",
  "exception": "RelationshipNotFoundException",
  "fullname": "org.neo4j.server.rest.web.RelationshipNotFoundException",
  "stackTrace": [
    "org.neo4j.server.rest.web.DatabaseActions.relationship(DatabaseActions.java:182)",
    "org.neo4j.server.rest.web.DatabaseActions.removeAllRelationshipProperties(DatabaseActions.java:653)",

"org.neo4j.server.rest.web.RestfulGraphDatabase.deleteAllRelationshipProperties(RestfulGraphDatabase.java:
761)",
    "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
    "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
    "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
    "java.lang.Thread.run(Thread.java:745)"
  ],
  "cause": {
    "exception": "NotFoundException",
    "cause": {
      "exception": "EntityNotFoundException",
      "fullname": "org.neo4j.internal.kernel.api.exceptions.EntityNotFoundException",
      "stackTrace": [

"org.neo4j.kernel.impl.factory.GraphDatabaseFacade.getRelationshipById(GraphDatabaseFacade.java:335)",
        "org.neo4j.server.rest.web.DatabaseActions.relationship(DatabaseActions.java:178)",

"org.neo4j.server.rest.web.DatabaseActions.removeAllRelationshipProperties(DatabaseActions.java:653)",

"org.neo4j.server.rest.web.RestfulGraphDatabase.deleteAllRelationshipProperties(RestfulGraphDatabase.java:
761)",
        "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
        "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
        "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
        "java.lang.Thread.run(Thread.java:745)"
      ],
      "message": "Unable to load RELATIONSHIP with id 1234.",
      "errors": [
        {
          "code": "Neo.ClientError.Statement.EntityNotFound",
          "message": "Unable to load RELATIONSHIP with id 1234."
        }
      ]
    },
    "fullname": "org.neo4j.graphdb.NotFoundException",
    "stackTrace": [

"org.neo4j.kernel.impl.factory.GraphDatabaseFacade.getRelationshipById(GraphDatabaseFacade.java:335)",
      "org.neo4j.server.rest.web.DatabaseActions.relationship(DatabaseActions.java:178)",

"org.neo4j.server.rest.web.DatabaseActions.removeAllRelationshipProperties(DatabaseActions.java:653)",

"org.neo4j.server.rest.web.RestfulGraphDatabase.deleteAllRelationshipProperties(RestfulGraphDatabase.java:
761)",
      "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
      "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
      "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
      "java.lang.Thread.run(Thread.java:745)"
    ],
    "message": "Relationship 1234 not found",
    "errors": [
      {
        "code": "Neo.ClientError.Statement.EntityNotFound",
        "message": "Relationship 1234 not found"
      }
    ]
  },
  "errors": [
    {
      "code": "Neo.ClientError.Statement.EntityNotFound",
      "message": "org.neo4j.graphdb.NotFoundException: Relationship 1234 not found"
    }
  ]
}
```

Remove property from a non-existing relationship

Attempting to remove a property from a relationship which doesn’t exist results in an error.

Example request

- **DELETE** http://localhost:7474/db/data/relationship/1234/properties/cost
- **Accept:** application/json; charset=UTF-8

Example response

- **404:** Not Found
- **Content-Type:** application/json;charset=utf-8

```
{
  "message": "org.neo4j.graphdb.NotFoundException: Relationship 1234 not found",
  "exception": "RelationshipNotFoundException",
  "fullname": "org.neo4j.server.rest.web.RelationshipNotFoundException",
  "stackTrace": [
    "org.neo4j.server.rest.web.DatabaseActions.relationship(DatabaseActions.java:182)",
    "org.neo4j.server.rest.web.DatabaseActions.removeRelationshipProperty(DatabaseActions.java:659)",

"org.neo4j.server.rest.web.RestfulGraphDatabase.deleteRelationshipProperty(RestfulGraphDatabase.java:781)"
,
    "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
    "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
    "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
    "java.lang.Thread.run(Thread.java:745)"
  ],
  "cause": {
    "exception": "NotFoundException",
    "cause": {
      "exception": "EntityNotFoundException",
      "fullname": "org.neo4j.internal.kernel.api.exceptions.EntityNotFoundException",
      "stackTrace": [

"org.neo4j.kernel.impl.factory.GraphDatabaseFacade.getRelationshipById(GraphDatabaseFacade.java:335)",
        "org.neo4j.server.rest.web.DatabaseActions.relationship(DatabaseActions.java:178)",
        "org.neo4j.server.rest.web.DatabaseActions.removeRelationshipProperty(DatabaseActions.java:659)",

"org.neo4j.server.rest.web.RestfulGraphDatabase.deleteRelationshipProperty(RestfulGraphDatabase.java:781)"
,
        "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
        "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
        "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
        "java.lang.Thread.run(Thread.java:745)"
      ],
      "message": "Unable to load RELATIONSHIP with id 1234.",
      "errors": [
        {
          "code": "Neo.ClientError.Statement.EntityNotFound",
          "message": "Unable to load RELATIONSHIP with id 1234."
        }
      ]
    },
    "fullname": "org.neo4j.graphdb.NotFoundException",
    "stackTrace": [

"org.neo4j.kernel.impl.factory.GraphDatabaseFacade.getRelationshipById(GraphDatabaseFacade.java:335)",
      "org.neo4j.server.rest.web.DatabaseActions.relationship(DatabaseActions.java:178)",
      "org.neo4j.server.rest.web.DatabaseActions.removeRelationshipProperty(DatabaseActions.java:659)",

"org.neo4j.server.rest.web.RestfulGraphDatabase.deleteRelationshipProperty(RestfulGraphDatabase.java:781)"
,
      "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
      "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
      "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
      "java.lang.Thread.run(Thread.java:745)"
    ],
    "message": "Relationship 1234 not found",
    "errors": [
      {
        "code": "Neo.ClientError.Statement.EntityNotFound",
        "message": "Relationship 1234 not found"
      }
    ]
  },
  "errors": [
    {
      "code": "Neo.ClientError.Statement.EntityNotFound",
      "message": "org.neo4j.graphdb.NotFoundException: Relationship 1234 not found"
    }
  ]
}
```

Node labels

Adding a label to a node

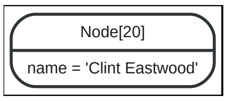


Figure 52. Starting Graph

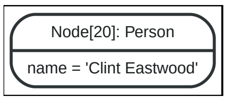


Figure 53. Final Graph

Example request

- **POST** http://localhost:7474/db/data/node/20/labels
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
"Person"
```

Example response

- **204:** No Content

```

```

Adding multiple labels to a node

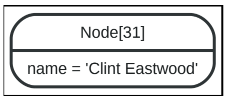


Figure 54. Starting Graph

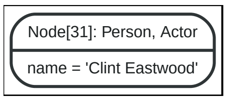


Figure 55. Final Graph

Example request

- **POST** http://localhost:7474/db/data/node/31/labels
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
[ "Person", "Actor" ]
```

Example response

- **204:** No Content

```
```

Adding a label with an invalid name

Labels with empty names are not allowed, however, all other valid strings are accepted as label names. Adding an invalid label to a node will lead to a HTTP 400 response.

Example request

- **POST** http://localhost:7474/db/data/node/38/labels
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
""
```

Example response

- **400:** Bad Request
- **Content-Type:** application/json;charset=utf-8

```
{
  "message": "Unable to add label, see nested exception.",
  "exception": "BadInputException",
  "fullname": "org.neo4j.server.rest.repr.BadInputException",
  "stackTrace": [
    "org.neo4j.server.rest.web.DatabaseActions.addLabelToNode(DatabaseActions.java:307)",
    "org.neo4j.server.rest.web.RestfulGraphDatabase.addNodeLabel(RestfulGraphDatabase.java:431)",
    "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
    "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
    "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
    "java.lang.Thread.run(Thread.java:745)"
  ],
  "cause": {
    "exception": "ConstraintViolationException",
    "cause": {
      "exception": "IllegalTokenNameException",
      "fullname": "org.neo4j.internal.kernel.api.exceptions.schema.IllegalTokenNameException",
      "stackTrace": [
        "org.neo4j.kernel.impl.newapi.KernelToken.checkValidTokenName(KernelToken.java:200)",
        "org.neo4j.kernel.impl.newapi.KernelToken.labelGetOrCreateForName(KernelToken.java:52)",
        "org.neo4j.kernel.impl.core.NodeProxy.addLabel(NodeProxy.java:574)",
        "org.neo4j.server.rest.web.DatabaseActions.addLabelToNode(DatabaseActions.java:302)",
        "org.neo4j.server.rest.web.RestfulGraphDatabase.addNodeLabel(RestfulGraphDatabase.java:431)",
        "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
        "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
        "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
        "java.lang.Thread.run(Thread.java:745)"
      ],
      "message": "\u0027\u0027 is not a valid token name. Only non-null, non-empty strings are allowed.",
      "errors": [
        {
          "code": "Neo.ClientError.Schema.TokenNameError",
          "message": "\u0027\u0027 is not a valid token name. Only non-null, non-empty strings are
allowed."
        }
      ]
    },
    "fullname": "org.neo4j.graphdb.ConstraintViolationException",
    "stackTrace": [
      "org.neo4j.kernel.impl.core.NodeProxy.addLabel(NodeProxy.java:583)",
      "org.neo4j.server.rest.web.DatabaseActions.addLabelToNode(DatabaseActions.java:302)",
      "org.neo4j.server.rest.web.RestfulGraphDatabase.addNodeLabel(RestfulGraphDatabase.java:431)",
      "java.lang.reflect.Method.invoke(Method.java:498)",

"org.neo4j.server.rest.transactional.TransactionalRequestDispatcher.dispatch(TransactionalRequestDispatche
r.java:147)",

"org.neo4j.server.rest.dbms.AuthorizationDisabledFilter.doFilter(AuthorizationDisabledFilter.java:49)",
      "org.neo4j.server.rest.web.CorsFilter.doFilter(CorsFilter.java:115)",
      "org.neo4j.server.rest.web.CollectUserAgentFilter.doFilter(CollectUserAgentFilter.java:69)",
      "java.lang.Thread.run(Thread.java:745)"
    ],
    "message": "Invalid label name \u0027\u0027.",
    "errors": [
      {
        "code": "Neo.ClientError.Schema.ConstraintValidationFailed",
        "message": "Invalid label name \u0027\u0027."
      }
    ]
  },
  "errors": [
    {
      "code": "Neo.ClientError.Request.InvalidFormat",
      "message": "Unable to add label, see nested exception."
    }
  ]
}
```

Replacing labels on a node

This removes any labels currently on a node, and replaces them with the labels passed in as the request body.

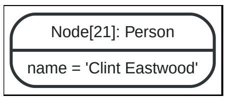


Figure 56. Starting Graph

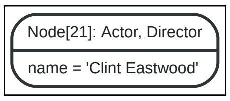


Figure 57. Final Graph

Example request

- **PUT** http://localhost:7474/db/data/node/21/labels
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
[ "Actor", "Director" ]
```

Example response

- **204:** No Content

```

```

Removing a label from a node

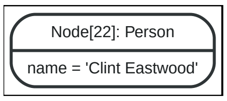


Figure 58. Starting Graph

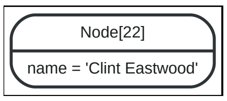


Figure 59. Final Graph

Example request

- **DELETE** http://localhost:7474/db/data/node/22/labels/Person
- **Accept:** application/json; charset=UTF-8

Example response

- **204:** No Content

```

```

Removing a non-existent label from a node

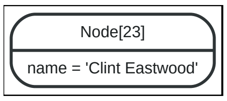


Figure 60. Starting Graph

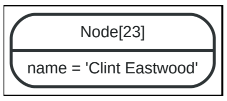


Figure 61. Final Graph

Example request

- **DELETE** http://localhost:7474/db/data/node/23/labels/Person
- **Accept:** application/json; charset=UTF-8

Example response

- **204:** No Content

```

```

Listing labels for a node

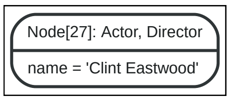


Figure 62. Final Graph

Example request

- **GET** http://localhost:7474/db/data/node/27/labels
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ "Actor", "Director" ]
```

Get all nodes with a label

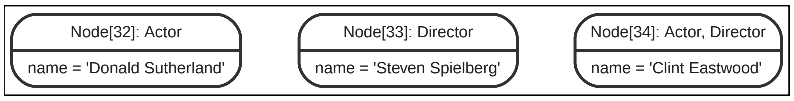

Node[32]: Actor
name = 'Donald Sutherland'
Node[33]: Director
name = 'Steven Spielberg'
Node[34]: Actor, Director
name = 'Clint Eastwood'

Figure 63. Final Graph

Example request

- **GET** http://localhost:7474/db/data/label/Actor/nodes
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "metadata" : {
    "id" : 32,
    "labels" : [ "Actor" ]
  },
  "data" : {
    "name" : "Donald Sutherland"
  },
  "paged_traverse" :
"http://localhost:7474/db/data/node/32/paged/traverse/{returnType}{?pageSize,leaseTime}",
  "outgoing_relationships" : "http://localhost:7474/db/data/node/32/relationships/out",
  "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/32/relationships/out/{-
list|&|types}",
  "create_relationship" : "http://localhost:7474/db/data/node/32/relationships",
  "labels" : "http://localhost:7474/db/data/node/32/labels",
  "traverse" : "http://localhost:7474/db/data/node/32/traverse/{returnType}",
  "extensions" : { },
  "all_relationships" : "http://localhost:7474/db/data/node/32/relationships/all",
  "all_typed_relationships" : "http://localhost:7474/db/data/node/32/relationships/all/{-list|&|types}",
  "property" : "http://localhost:7474/db/data/node/32/properties/{key}",
  "self" : "http://localhost:7474/db/data/node/32",
  "incoming_relationships" : "http://localhost:7474/db/data/node/32/relationships/in",
  "properties" : "http://localhost:7474/db/data/node/32/properties",
  "incoming_typed_relationships" : "http://localhost:7474/db/data/node/32/relationships/in/{-
list|&|types}"
}, {
  "metadata" : {
    "id" : 34,
    "labels" : [ "Actor", "Director" ]
  },
  "data" : {
    "name" : "Clint Eastwood"
  },
  "paged_traverse" :
"http://localhost:7474/db/data/node/34/paged/traverse/{returnType}{?pageSize,leaseTime}",
  "outgoing_relationships" : "http://localhost:7474/db/data/node/34/relationships/out",
  "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/34/relationships/out/{-
list|&|types}",
  "create_relationship" : "http://localhost:7474/db/data/node/34/relationships",
  "labels" : "http://localhost:7474/db/data/node/34/labels",
  "traverse" : "http://localhost:7474/db/data/node/34/traverse/{returnType}",
  "extensions" : { },
  "all_relationships" : "http://localhost:7474/db/data/node/34/relationships/all",
  "all_typed_relationships" : "http://localhost:7474/db/data/node/34/relationships/all/{-list|&|types}",
  "property" : "http://localhost:7474/db/data/node/34/properties/{key}",
  "self" : "http://localhost:7474/db/data/node/34",
  "incoming_relationships" : "http://localhost:7474/db/data/node/34/relationships/in",
  "properties" : "http://localhost:7474/db/data/node/34/properties",
  "incoming_typed_relationships" : "http://localhost:7474/db/data/node/34/relationships/in/{-
list|&|types}"
} ]
```

Get nodes by label and property

You can retrieve all nodes with a given label and property by passing one property as a query parameter. Notice that the property value is JSON-encoded and then URL-encoded.

If there is an index available on the label/property combination you send, that index will be used. If no index is available, all nodes with the given label will be filtered through to find matching nodes.

Currently, it is not possible to search using multiple properties.

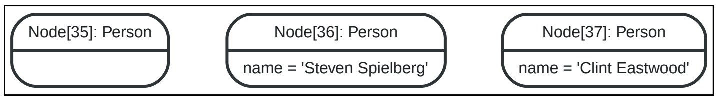


Figure 64. Final Graph

Example request

- **GET** http://localhost:7474/db/data/label/Person/nodes?name=%22Clint+Eastwood%22
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "metadata" : {
    "id" : 37,
    "labels" : [ "Person" ]
  },
  "data" : {
    "name" : "Clint Eastwood"
  },
  "paged_traverse" :
"http://localhost:7474/db/data/node/37/paged/traverse/{returnType}{?pageSize,leaseTime}",
  "outgoing_relationships" : "http://localhost:7474/db/data/node/37/relationships/out",
  "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/37/relationships/out/{-
list|&|types}",
  "create_relationship" : "http://localhost:7474/db/data/node/37/relationships",
  "labels" : "http://localhost:7474/db/data/node/37/labels",
  "traverse" : "http://localhost:7474/db/data/node/37/traverse/{returnType}",
  "extensions" : { },
  "all_relationships" : "http://localhost:7474/db/data/node/37/relationships/all",
  "all_typed_relationships" : "http://localhost:7474/db/data/node/37/relationships/all/{-list|&|types}",
  "property" : "http://localhost:7474/db/data/node/37/properties/{key}",
  "self" : "http://localhost:7474/db/data/node/37",
  "incoming_relationships" : "http://localhost:7474/db/data/node/37/relationships/in",
  "properties" : "http://localhost:7474/db/data/node/37/properties",
  "incoming_typed_relationships" : "http://localhost:7474/db/data/node/37/relationships/in/{-
list|&|types}"
} ]
```

List all labels

By default, the server will return labels in use only. If you also want to return labels not in use, append the "in_use=0" query parameter.

Example request

- **GET** http://localhost:7474/db/data/labels
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ "Person", "Director", "Actor" ]
```

Node degree

The node degree is the number of relationships associated with a node. Neo4j stores the degree for each node, making this a useful mechanism to quickly get the number of relationships a node has. You can also optionally filter degree by direction and/or relationship type.

Get the degree of a node

Return the total number of relationships associated with a node.

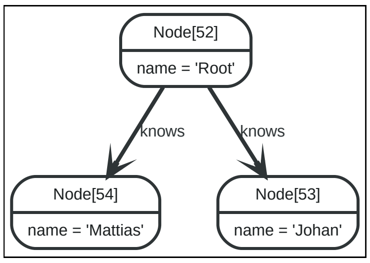


Figure 65. Final Graph

Example request

- **GET** http://localhost:7474/db/data/node/52/degree/all
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
2
```

Get the degree of a node by direction

Return the number of relationships of a particular direction for a node. Specify `all`, `in` or `out`.

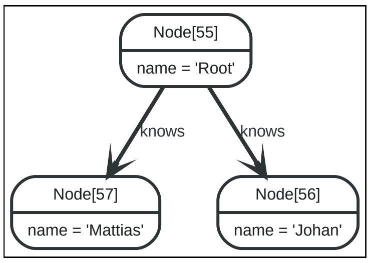


Figure 66. Final Graph

Example request

- **GET** http://localhost:7474/db/data/node/55/degree/out
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
2
```

Get the degree of a node by direction and types

If you are only interested in the degree of a particular relationship type, or a set of relationship types, you specify relationship types after the direction. You can combine multiple relationship types by using the `&` character.

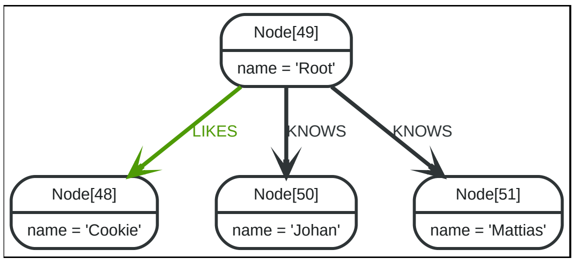


Figure 67. Final Graph

Example request

- **GET** http://localhost:7474/db/data/node/49/degree/out/KNOWS&LIKES
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
3
```

Indexing

How to use schema based indexes from the REST API. For more details about indexes and the optional schema in Neo4j, see the Getting Started Guide.

For how to use explicit indexes, see Explicit indexing.

Create index

This will start a background job in the database that will create and populate the index. You can check the status of your index by listing all the indexes for the relevant label.

Example request

- **POST** http://localhost:7474/db/data/schema/index/label_1528894885115_1
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "property_keys" : [ "property_1528894885115_1" ]
}
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "label" : "label_1528894885115_1",
  "property_keys" : [ "property_1528894885115_1" ]
}
```

List indexes for a label

Example request

- **GET** http://localhost:7474/db/data/schema/index/label_1528894884999_1
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "property_keys" : [ "property_1528894884999_1" ],
  "label" : "label_1528894884999_1"
} ]
```

Drop index

Drop index

Example request

- **DELETE** http://localhost:7474/db/data/schema/index/label_1528894885062_1/property_1528894885062_1
- **Accept:** application/json; charset=UTF-8

Example response

- **204:** No Content

```

```

Constraints

Create uniqueness constraint

Create a uniqueness constraint on a property.

Example request

- **POST** http://localhost:7474/db/data/schema/constraint/label_1528894851824_1/uniqueness/
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "property_keys" : [ "property_1528894851824_1" ]
}
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "label" : "label_1528894851824_1",
  "type" : "UNIQUENESS",
  "property_keys" : [ "property_1528894851824_1" ]
}
```

Get a specific uniqueness constraint

Get a specific uniqueness constraint for a label and a property.

Example request

- **GET** http://localhost:7474/db/data/schema/constraint/label_1528894852104_1/uniqueness/property_1528894852104_1
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "property_keys" : [ "property_1528894852104_1" ],
  "label" : "label_1528894852104_1",
  "type" : "UNIQUENESS"
} ]
```

Get all uniqueness constraints for a label

Example request

- **GET** http://localhost:7474/db/data/schema/constraint/label_1528894852027_1/uniqueness/
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "property_keys" : [ "property_1528894852027_2" ],
  "label" : "label_1528894852027_1",
  "type" : "UNIQUENESS"
}, {
  "property_keys" : [ "property_1528894852027_1" ],
  "label" : "label_1528894852027_1",
  "type" : "UNIQUENESS"
} ]
```

Drop uniqueness constraint

Drop uniqueness constraint for a label and a property.

Example request

- **DELETE** http://localhost:7474/db/data/schema/constraint/label_1528894851612_1/uniqueness/property_1528894851612_1
- **Accept:** application/json; charset=UTF-8

Example response

- **204:** No Content

```

```

Get a specific node property existence constraint

Get a specific node property existence constraint for a label and a property.

Example request

- **GET** http://localhost:7474/db/data/schema/constraint/label_1528891028379_1/existence/property_1528891028379_1
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "property_keys" : [ "property_1528891028379_1" ],
  "label" : "label_1528891028379_1",
  "type" : "NODE_PROPERTY_EXISTENCE"
} ]
```

Get all node property existence constraints for a label

Example request

- **GET** http://localhost:7474/db/data/schema/constraint/label_1528891028458_1/existence/
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "property_keys" : [ "property_1528891028458_2" ],
  "label" : "label_1528891028458_1",
  "type" : "NODE_PROPERTY_EXISTENCE"
}, {
  "property_keys" : [ "property_1528891028458_1" ],
  "label" : "label_1528891028458_1",
  "type" : "NODE_PROPERTY_EXISTENCE"
} ]
```

Get all constraints for a label

Example request

- **GET** http://localhost:7474/db/data/schema/constraint/label_1528894851950_1
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "property_keys" : [ "property_1528894851950_1" ],
  "label" : "label_1528894851950_1",
  "type" : "UNIQUENESS"
} ]
```

Get a specific relationship property existence constraint

Get a specific relationship property existence constraint for a label and a property.

Example request

- **GET** http://localhost:7474/db/data/schema/relationship/constraint/relationshipType_1528891025070_1/existence/property_1528891025070_1
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "relationshipType" : "relationshipType_1528891025070_1",
  "property_keys" : [ "property_1528891025070_1" ],
  "type" : "RELATIONSHIP_PROPERTY_EXISTENCE"
} ]
```

Get all relationship property existence constraints for a type

Example request

- **GET** http://localhost:7474/db/data/schema/relationship/constraint/relationshipType_1528891028272_1/existence/
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "relationshipType" : "relationshipType_1528891028272_1",
  "property_keys" : [ "property_1528891028272_2" ],
  "type" : "RELATIONSHIP_PROPERTY_EXISTENCE"
}, {
  "relationshipType" : "relationshipType_1528891028272_1",
  "property_keys" : [ "property_1528891028272_1" ],
  "type" : "RELATIONSHIP_PROPERTY_EXISTENCE"
} ]
```

Get all constraints

Example request

- **GET** http://localhost:7474/db/data/schema/constraint
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "property_keys" : [ "property_1528894851824_1" ],
  "label" : "label_1528894851824_1",
  "type" : "UNIQUENESS"
}, {
  "property_keys" : [ "property_1528894851886_1" ],
  "label" : "label_1528894851886_1",
  "type" : "UNIQUENESS"
} ]
```

Graph Algorithms

Neo4j comes with a number of built-in graph algorithms. They are performed from a start node. The traversal is controlled by the URI and the body sent with the request. These are the parameters that can be used:

algorithm

The algorithm to choose. If not set, default is `shortestPath`. `algorithm` can have one of these values:

- `shortestPath`
- `allSimplePaths`
- `allPaths`
- `dijkstra` (optionally with `cost_property` and `default_cost` parameters)

max_depth

The maximum depth as an integer for the algorithms like `shortestPath`, where applicable. Default is `1`.

Find all shortest paths

The shortestPath algorithm can find multiple paths between the same nodes, like in this example.

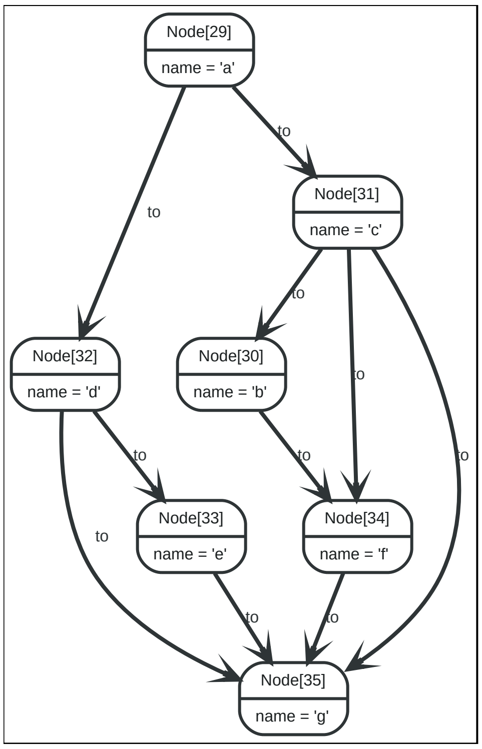


Figure 68. Final Graph

Example request

- **POST** http://localhost:7474/db/data/node/29/paths
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "to" : "http://localhost:7474/db/data/node/35",
  "max_depth" : 3,
  "relationships" : {
    "type" : "to",
    "direction" : "out"
  },
  "algorithm" : "shortestPath"
}
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "relationships" : [ "http://localhost:7474/db/data/relationship/28",
"http://localhost:7474/db/data/relationship/34" ],
  "nodes" : [ "http://localhost:7474/db/data/node/29", "http://localhost:7474/db/data/node/32",
"http://localhost:7474/db/data/node/35" ],
  "directions" : [ "->", "->" ],
  "length" : 2,
  "start" : "http://localhost:7474/db/data/node/29",
  "end" : "http://localhost:7474/db/data/node/35"
}, {
  "relationships" : [ "http://localhost:7474/db/data/relationship/27",
"http://localhost:7474/db/data/relationship/36" ],
  "nodes" : [ "http://localhost:7474/db/data/node/29", "http://localhost:7474/db/data/node/31",
"http://localhost:7474/db/data/node/35" ],
  "directions" : [ "->", "->" ],
  "length" : 2,
  "start" : "http://localhost:7474/db/data/node/29",
  "end" : "http://localhost:7474/db/data/node/35"
} ]
```

Find one of the shortest paths

If no path algorithm is specified, a shortestPath algorithm with a max depth of 1 will be chosen. In this example, the max_depth is set to 3 in order to find the shortest path between a maximum of 3 linked nodes.

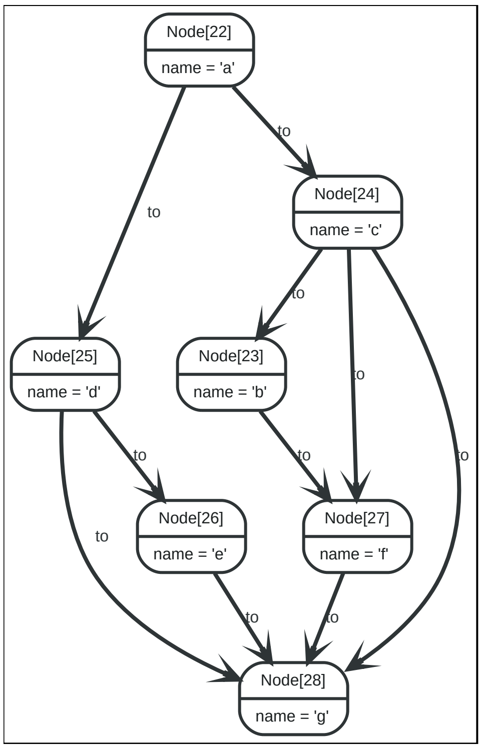


Figure 69. Final Graph

Example request

- **POST** http://localhost:7474/db/data/node/22/path
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "to" : "http://localhost:7474/db/data/node/28",
  "max_depth" : 3,
  "relationships" : {
    "type" : "to",
    "direction" : "out"
  },
  "algorithm" : "shortestPath"
}
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "relationships" : [ "http://localhost:7474/db/data/relationship/17",
"http://localhost:7474/db/data/relationship/26" ],
  "nodes" : [ "http://localhost:7474/db/data/node/22", "http://localhost:7474/db/data/node/24",
"http://localhost:7474/db/data/node/28" ],
  "directions" : [ "->", "->" ],
  "length" : 2,
  "start" : "http://localhost:7474/db/data/node/22",
  "end" : "http://localhost:7474/db/data/node/28"
}
```

Execute a Dijkstra algorithm and get a single path

This example is running a Dijkstra algorithm over a graph with different cost properties on different relationships. Note that the request URI ends with /path which means a single path is what we want here.

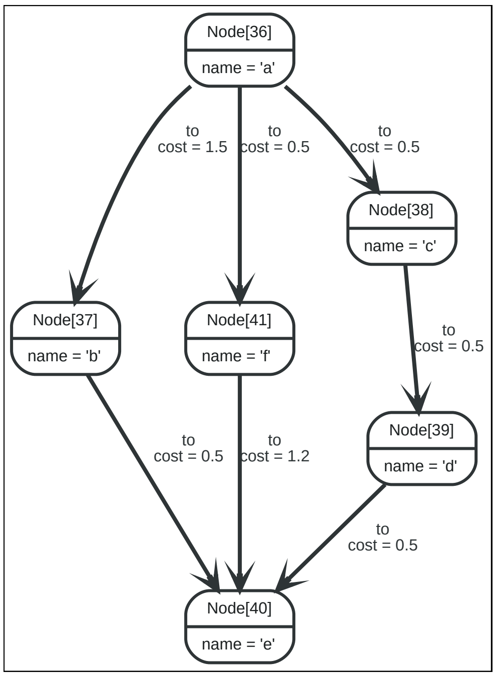


Figure 70. Final Graph

Example request

- **POST** http://localhost:7474/db/data/node/36/path
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "to" : "http://localhost:7474/db/data/node/40",
  "cost_property" : "cost",
  "relationships" : {
    "type" : "to",
    "direction" : "out"
  },
  "algorithm" : "dijkstra"
}
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "relationships" : [ "http://localhost:7474/db/data/relationship/38",
"http://localhost:7474/db/data/relationship/40", "http://localhost:7474/db/data/relationship/41" ],
  "nodes" : [ "http://localhost:7474/db/data/node/36", "http://localhost:7474/db/data/node/38",
"http://localhost:7474/db/data/node/39", "http://localhost:7474/db/data/node/40" ],
  "directions" : [ "->", "->", "->" ],
  "length" : 3,
  "start" : "http://localhost:7474/db/data/node/36",
  "weight" : 1.5,
  "end" : "http://localhost:7474/db/data/node/40"
}
```

Execute a Dijkstra algorithm with equal weights on relationships

The following is executing a Dijkstra search on a graph with equal weights on all relationships. This example is included to show the difference when the same graph structure is used, but the path weight is equal to the number of hops.

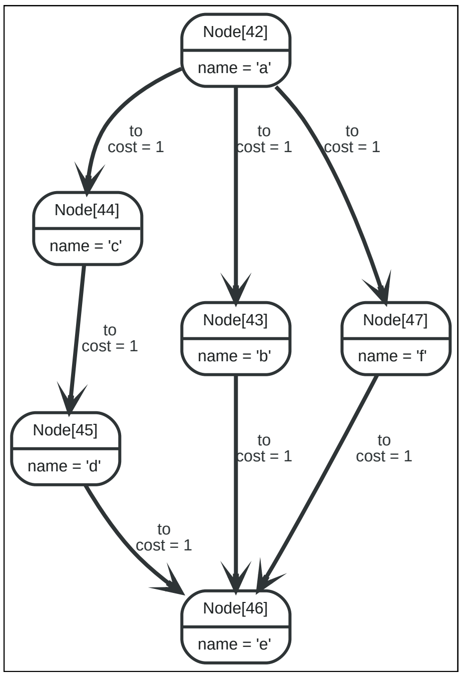


Figure 71. Final Graph

Example request

- **POST** http://localhost:7474/db/data/node/42/path
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "to" : "http://localhost:7474/db/data/node/46",
  "cost_property" : "cost",
  "relationships" : {
    "type" : "to",
    "direction" : "out"
  },
  "algorithm" : "dijkstra"
}
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "relationships" : [ "http://localhost:7474/db/data/relationship/46",
"http://localhost:7474/db/data/relationship/50" ],
  "nodes" : [ "http://localhost:7474/db/data/node/42", "http://localhost:7474/db/data/node/47",
"http://localhost:7474/db/data/node/46" ],
  "directions" : [ "->", "->" ],
  "length" : 2,
  "start" : "http://localhost:7474/db/data/node/42",
  "weight" : 2.0,
  "end" : "http://localhost:7474/db/data/node/46"
}
```

Execute a Dijkstra algorithm and get multiple paths

This example is running a Dijkstra algorithm over a graph with different cost properties on different relationships. Note that the request URI ends with /paths which means we want multiple paths returned, in case they exist.

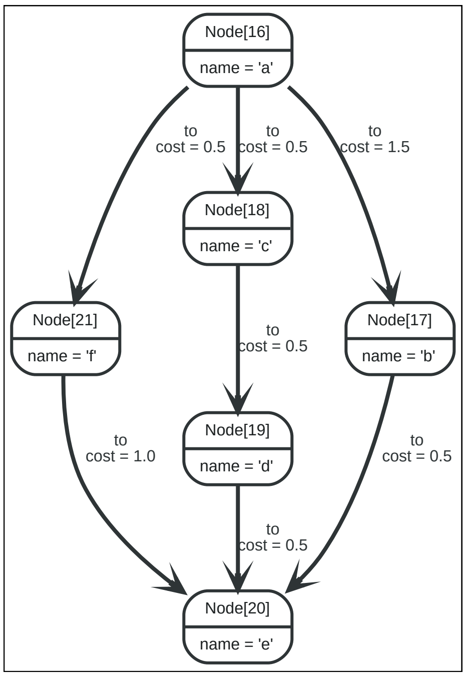


Figure 72. Final Graph

Example request

- **POST** http://localhost:7474/db/data/node/16/paths
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "to" : "http://localhost:7474/db/data/node/20",
  "cost_property" : "cost",
  "relationships" : {
    "type" : "to",
    "direction" : "out"
  },
  "algorithm" : "dijkstra"
}
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "relationships" : [ "http://localhost:7474/db/data/relationship/12",
"http://localhost:7474/db/data/relationship/16" ],
  "nodes" : [ "http://localhost:7474/db/data/node/16", "http://localhost:7474/db/data/node/21",
"http://localhost:7474/db/data/node/20" ],
  "directions" : [ "->", "->" ],
  "length" : 2,
  "start" : "http://localhost:7474/db/data/node/16",
  "weight" : 1.5,
  "end" : "http://localhost:7474/db/data/node/20"
}, {
  "relationships" : [ "http://localhost:7474/db/data/relationship/11",
"http://localhost:7474/db/data/relationship/13", "http://localhost:7474/db/data/relationship/14" ],
  "nodes" : [ "http://localhost:7474/db/data/node/16", "http://localhost:7474/db/data/node/18",
"http://localhost:7474/db/data/node/19", "http://localhost:7474/db/data/node/20" ],
  "directions" : [ "->", "->", "->" ],
  "length" : 3,
  "start" : "http://localhost:7474/db/data/node/16",
  "weight" : 1.5,
  "end" : "http://localhost:7474/db/data/node/20"
} ]
```

Batch operations

Batch operations lets you execute multiple API calls through a single HTTP call. This improves performance for large insert and update operations significantly.

This service is *transactional*. If any of the operations performed fails (returns a non-2xx HTTP status code), the transaction will be rolled back and no changes will be applied.

IMPORTANT:

You cannot use this resource to execute Cypher queries with `USING PERIODIC COMMIT`.

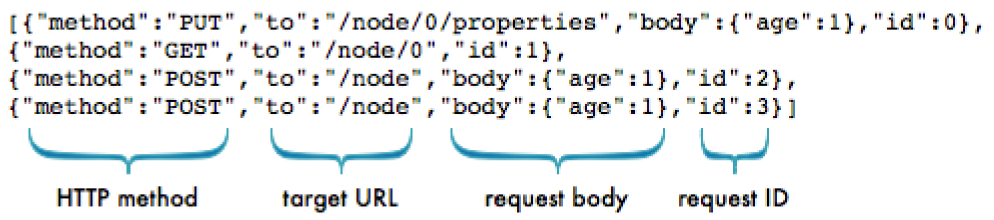
Execute multiple operations in batch

The batch service expects an array of job descriptions as input, each job description describing an action to be performed via the normal server API.

Each job description should contain a `to` attribute, with a value relative to the data API root (so `http://localhost:7474/db/data/node` becomes just `/node`), and a `method` attribute containing HTTP verb to use.

Optionally you may provide a `body` attribute, and an `id` attribute to help you keep track of responses, although responses are guaranteed to be returned in the same order the job descriptions are received.

The following figure outlines the different parts of the job descriptions:



```
[{"method":"PUT","to":"/node/0/properties","body":{"age":1},"id":0},
{"method":"GET","to":"/node/0","id":1},
{"method":"POST","to":"/node","body":{"age":1},"id":2},
{"method":"POST","to":"/node","body":{"age":1},"id":3}]
```

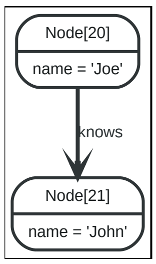


Figure 73. Starting Graph

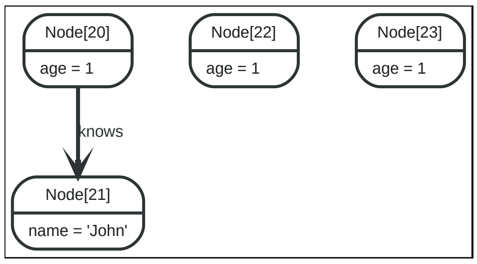


Figure 74. Final Graph

Example request

- **POST** http://localhost:7474/db/data/batch
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
[ {
  "method" : "PUT",
  "to" : "/node/20/properties",
  "body" : {
    "age" : 1
  },
  "id" : 0
}, {
  "method" : "GET",
  "to" : "/node/20",
  "id" : 1
}, {
  "method" : "POST",
  "to" : "/node",
  "body" : {
    "age" : 1
  },
  "id" : 2
}, {
  "method" : "POST",
  "to" : "/node",
  "body" : {
    "age" : 1
  },
  "id" : 3
} ]
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "id" : 0,
  "from" : "/node/20/properties"
}, {
  "id" : 1,
  "body" : {
    "extensions" : { },
    "metadata" : {
      "id" : 20,
      "labels" : [ ]
    },
    "paged_traverse" :
"http://localhost:7474/db/data/node/20/paged/traverse/{returnType}{?pageSize,leaseTime}",
    "outgoing_relationships" : "http://localhost:7474/db/data/node/20/relationships/out",
    "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/20/relationships/out/{-
list|&|types}",
    "create_relationship" : "http://localhost:7474/db/data/node/20/relationships",
    "labels" : "http://localhost:7474/db/data/node/20/labels",
    "traverse" : "http://localhost:7474/db/data/node/20/traverse/{returnType}",
    "all_relationships" : "http://localhost:7474/db/data/node/20/relationships/all",
    "all_typed_relationships" : "http://localhost:7474/db/data/node/20/relationships/all/{-list|&|types}",
    "property" : "http://localhost:7474/db/data/node/20/properties/{key}",
    "self" : "http://localhost:7474/db/data/node/20",
    "incoming_relationships" : "http://localhost:7474/db/data/node/20/relationships/in",
    "properties" : "http://localhost:7474/db/data/node/20/properties",
    "incoming_typed_relationships" : "http://localhost:7474/db/data/node/20/relationships/in/{-
list|&|types}",
    "data" : {
      "age" : 1
    }
  },
  "from" : "/node/20"
}, {
  "id" : 2,
  "location" : "http://localhost:7474/db/data/node/22",
  "body" : {
    "extensions" : { },
    "metadata" : {
      "id" : 22,
      "labels" : [ ]
    },
    "paged_traverse" :
"http://localhost:7474/db/data/node/22/paged/traverse/{returnType}{?pageSize,leaseTime}",
    "outgoing_relationships" : "http://localhost:7474/db/data/node/22/relationships/out",
    "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/22/relationships/out/{-
list|&|types}",
    "create_relationship" : "http://localhost:7474/db/data/node/22/relationships",
    "labels" : "http://localhost:7474/db/data/node/22/labels",
    "traverse" : "http://localhost:7474/db/data/node/22/traverse/{returnType}",
    "all_relationships" : "http://localhost:7474/db/data/node/22/relationships/all",
    "all_typed_relationships" : "http://localhost:7474/db/data/node/22/relationships/all/{-list|&|types}",
    "property" : "http://localhost:7474/db/data/node/22/properties/{key}",
    "self" : "http://localhost:7474/db/data/node/22",
    "incoming_relationships" : "http://localhost:7474/db/data/node/22/relationships/in",
    "properties" : "http://localhost:7474/db/data/node/22/properties",
    "incoming_typed_relationships" : "http://localhost:7474/db/data/node/22/relationships/in/{-
list|&|types}",
    "data" : {
      "age" : 1
    }
  },
  "from" : "/node"
}, {
  "id" : 3,
  "location" : "http://localhost:7474/db/data/node/23",
  "body" : {
    "extensions" : { },
    "metadata" : {
      "id" : 23,
      "labels" : [ ]
    },
    "paged_traverse" :
"http://localhost:7474/db/data/node/23/paged/traverse/{returnType}{?pageSize,leaseTime}",
    "outgoing_relationships" : "http://localhost:7474/db/data/node/23/relationships/out",
    "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/23/relationships/out/{-
list|&|types}",
    "create_relationship" : "http://localhost:7474/db/data/node/23/relationships",
    "labels" : "http://localhost:7474/db/data/node/23/labels",
    "traverse" : "http://localhost:7474/db/data/node/23/traverse/{returnType}",
    "all_relationships" : "http://localhost:7474/db/data/node/23/relationships/all",
    "all_typed_relationships" : "http://localhost:7474/db/data/node/23/relationships/all/{-list|&|types}",
    "property" : "http://localhost:7474/db/data/node/23/properties/{key}",
    "self" : "http://localhost:7474/db/data/node/23",
    "incoming_relationships" : "http://localhost:7474/db/data/node/23/relationships/in",
    "properties" : "http://localhost:7474/db/data/node/23/properties",
    "incoming_typed_relationships" : "http://localhost:7474/db/data/node/23/relationships/in/{-
list|&|types}",
    "data" : {
      "age" : 1
    }
  },
  "from" : "/node"
} ]
```

Refer to items created earlier in the same batch job

The batch operation API allows you to refer to the URI returned from a created resource in subsequent job descriptions, within the same batch call.

Use the `{[JOB ID]}` special syntax to inject URIs from created resources into JSON strings in subsequent job descriptions.

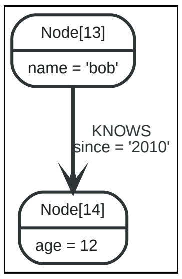


Figure 75. Final Graph

Example request

- **POST** http://localhost:7474/db/data/batch
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
[ {
  "method" : "POST",
  "to" : "/node",
  "id" : 0,
  "body" : {
    "name" : "bob"
  }
}, {
  "method" : "POST",
  "to" : "/node",
  "id" : 1,
  "body" : {
    "age" : 12
  }
}, {
  "method" : "POST",
  "to" : "{0}/relationships",
  "id" : 3,
  "body" : {
    "to" : "{1}",
    "data" : {
      "since" : "2010"
    },
    "type" : "KNOWS"
  }
}, {
  "method" : "POST",
  "to" : "/index/relationship/my_rels",
  "id" : 4,
  "body" : {
    "key" : "since",
    "value" : "2010",
    "uri" : "{3}"
  }
} ]
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "id" : 0,
  "location" : "http://localhost:7474/db/data/node/13",
  "body" : {
    "extensions" : { },
    "metadata" : {
      "id" : 13,
      "labels" : [ ]
    },
    "paged_traverse" :
"http://localhost:7474/db/data/node/13/paged/traverse/{returnType}{?pageSize,leaseTime}",
    "outgoing_relationships" : "http://localhost:7474/db/data/node/13/relationships/out",
    "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/13/relationships/out/{-
list|&|types}",
    "create_relationship" : "http://localhost:7474/db/data/node/13/relationships",
    "labels" : "http://localhost:7474/db/data/node/13/labels",
    "traverse" : "http://localhost:7474/db/data/node/13/traverse/{returnType}",
    "all_relationships" : "http://localhost:7474/db/data/node/13/relationships/all",
    "all_typed_relationships" : "http://localhost:7474/db/data/node/13/relationships/all/{-list|&|types}",
    "property" : "http://localhost:7474/db/data/node/13/properties/{key}",
    "self" : "http://localhost:7474/db/data/node/13",
    "incoming_relationships" : "http://localhost:7474/db/data/node/13/relationships/in",
    "properties" : "http://localhost:7474/db/data/node/13/properties",
    "incoming_typed_relationships" : "http://localhost:7474/db/data/node/13/relationships/in/{-
list|&|types}",
    "data" : {
      "name" : "bob"
    }
  },
  "from" : "/node"
}, {
  "id" : 1,
  "location" : "http://localhost:7474/db/data/node/14",
  "body" : {
    "extensions" : { },
    "metadata" : {
      "id" : 14,
      "labels" : [ ]
    },
    "paged_traverse" :
"http://localhost:7474/db/data/node/14/paged/traverse/{returnType}{?pageSize,leaseTime}",
    "outgoing_relationships" : "http://localhost:7474/db/data/node/14/relationships/out",
    "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/14/relationships/out/{-
list|&|types}",
    "create_relationship" : "http://localhost:7474/db/data/node/14/relationships",
    "labels" : "http://localhost:7474/db/data/node/14/labels",
    "traverse" : "http://localhost:7474/db/data/node/14/traverse/{returnType}",
    "all_relationships" : "http://localhost:7474/db/data/node/14/relationships/all",
    "all_typed_relationships" : "http://localhost:7474/db/data/node/14/relationships/all/{-list|&|types}",
    "property" : "http://localhost:7474/db/data/node/14/properties/{key}",
    "self" : "http://localhost:7474/db/data/node/14",
    "incoming_relationships" : "http://localhost:7474/db/data/node/14/relationships/in",
    "properties" : "http://localhost:7474/db/data/node/14/properties",
    "incoming_typed_relationships" : "http://localhost:7474/db/data/node/14/relationships/in/{-
list|&|types}",
    "data" : {
      "age" : 12
    }
  },
  "from" : "/node"
}, {
  "id" : 3,
  "location" : "http://localhost:7474/db/data/relationship/4",
  "body" : {
    "extensions" : { },
    "metadata" : {
      "id" : 4,
      "type" : "KNOWS"
    },
    "property" : "http://localhost:7474/db/data/relationship/4/properties/{key}",
    "start" : "http://localhost:7474/db/data/node/13",
    "self" : "http://localhost:7474/db/data/relationship/4",
    "end" : "http://localhost:7474/db/data/node/14",
    "type" : "KNOWS",
    "properties" : "http://localhost:7474/db/data/relationship/4/properties",
    "data" : {
      "since" : "2010"
    }
  },
  "from" : "http://localhost:7474/db/data/node/13/relationships"
}, {
  "id" : 4,
  "location" : "http://localhost:7474/db/data/index/relationship/my_rels/since/2010/4",
  "body" : {
    "extensions" : { },
    "metadata" : {
      "id" : 4,
      "type" : "KNOWS"
    },
    "property" : "http://localhost:7474/db/data/relationship/4/properties/{key}",
    "start" : "http://localhost:7474/db/data/node/13",
    "self" : "http://localhost:7474/db/data/relationship/4",
    "end" : "http://localhost:7474/db/data/node/14",
    "type" : "KNOWS",
    "properties" : "http://localhost:7474/db/data/relationship/4/properties",
    "data" : {
      "since" : "2010"
    },
    "indexed" : "http://localhost:7474/db/data/index/relationship/my_rels/since/2010/4"
  },
  "from" : "/index/relationship/my_rels"
} ]
```

Execute multiple operations in batch streaming

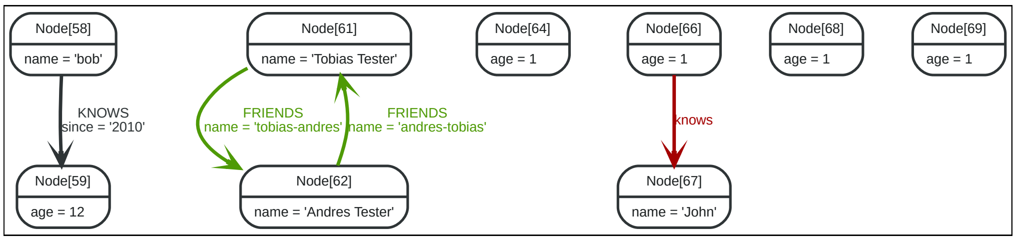


Figure 76. Final Graph

Example request

- **POST** http://localhost:7474/db/data/batch
- **Accept:** application/json
- **Content-Type:** application/json
- **X-Stream:** true

```
[ {
  "method" : "PUT",
  "to" : "/node/66/properties",
  "body" : {
    "age" : 1
  },
  "id" : 0
}, {
  "method" : "GET",
  "to" : "/node/66",
  "id" : 1
}, {
  "method" : "POST",
  "to" : "/node",
  "body" : {
    "age" : 1
  },
  "id" : 2
}, {
  "method" : "POST",
  "to" : "/node",
  "body" : {
    "age" : 1
  },
  "id" : 3
} ]
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "id" : 0,
  "from" : "/node/66/properties",
  "body" : null,
  "status" : 204
}, {
  "id" : 1,
  "from" : "/node/66",
  "body" : {
    "extensions" : { },
    "metadata" : {
      "id" : 66,
      "labels" : [ ]
    },
    "paged_traverse" :
"http://localhost:7474/db/data/node/66/paged/traverse/{returnType}{?pageSize,leaseTime}",
    "outgoing_relationships" : "http://localhost:7474/db/data/node/66/relationships/out",
    "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/66/relationships/out/{-
list|&|types}",
    "create_relationship" : "http://localhost:7474/db/data/node/66/relationships",
    "labels" : "http://localhost:7474/db/data/node/66/labels",
    "traverse" : "http://localhost:7474/db/data/node/66/traverse/{returnType}",
    "all_relationships" : "http://localhost:7474/db/data/node/66/relationships/all",
    "all_typed_relationships" : "http://localhost:7474/db/data/node/66/relationships/all/{-list|&|types}",
    "property" : "http://localhost:7474/db/data/node/66/properties/{key}",
    "self" : "http://localhost:7474/db/data/node/66",
    "incoming_relationships" : "http://localhost:7474/db/data/node/66/relationships/in",
    "properties" : "http://localhost:7474/db/data/node/66/properties",
    "incoming_typed_relationships" : "http://localhost:7474/db/data/node/66/relationships/in/{-
list|&|types}",
    "data" : {
      "age" : 1
    }
  },
  "status" : 200
}, {
  "id" : 2,
  "from" : "/node",
  "body" : {
    "extensions" : { },
    "metadata" : {
      "id" : 68,
      "labels" : [ ]
    },
    "paged_traverse" :
"http://localhost:7474/db/data/node/68/paged/traverse/{returnType}{?pageSize,leaseTime}",
    "outgoing_relationships" : "http://localhost:7474/db/data/node/68/relationships/out",
    "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/68/relationships/out/{-
list|&|types}",
    "create_relationship" : "http://localhost:7474/db/data/node/68/relationships",
    "labels" : "http://localhost:7474/db/data/node/68/labels",
    "traverse" : "http://localhost:7474/db/data/node/68/traverse/{returnType}",
    "all_relationships" : "http://localhost:7474/db/data/node/68/relationships/all",
    "all_typed_relationships" : "http://localhost:7474/db/data/node/68/relationships/all/{-list|&|types}",
    "property" : "http://localhost:7474/db/data/node/68/properties/{key}",
    "self" : "http://localhost:7474/db/data/node/68",
    "incoming_relationships" : "http://localhost:7474/db/data/node/68/relationships/in",
    "properties" : "http://localhost:7474/db/data/node/68/properties",
    "incoming_typed_relationships" : "http://localhost:7474/db/data/node/68/relationships/in/{-
list|&|types}",
    "data" : {
      "age" : 1
    }
  },
  "location" : "http://localhost:7474/db/data/node/68",
  "status" : 201
}, {
  "id" : 3,
  "from" : "/node",
  "body" : {
    "extensions" : { },
    "metadata" : {
      "id" : 69,
      "labels" : [ ]
    },
    "paged_traverse" :
"http://localhost:7474/db/data/node/69/paged/traverse/{returnType}{?pageSize,leaseTime}",
    "outgoing_relationships" : "http://localhost:7474/db/data/node/69/relationships/out",
    "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/69/relationships/out/{-
list|&|types}",
    "create_relationship" : "http://localhost:7474/db/data/node/69/relationships",
    "labels" : "http://localhost:7474/db/data/node/69/labels",
    "traverse" : "http://localhost:7474/db/data/node/69/traverse/{returnType}",
    "all_relationships" : "http://localhost:7474/db/data/node/69/relationships/all",
    "all_typed_relationships" : "http://localhost:7474/db/data/node/69/relationships/all/{-list|&|types}",
    "property" : "http://localhost:7474/db/data/node/69/properties/{key}",
    "self" : "http://localhost:7474/db/data/node/69",
    "incoming_relationships" : "http://localhost:7474/db/data/node/69/relationships/in",
    "properties" : "http://localhost:7474/db/data/node/69/properties",
    "incoming_typed_relationships" : "http://localhost:7474/db/data/node/69/relationships/in/{-
list|&|types}",
    "data" : {
      "age" : 1
    }
  },
  "location" : "http://localhost:7474/db/data/node/69",
  "status" : 201
} ]
```

Explicit indexing

This documents the explicit indexing in Neo4j, which is deprecated for removal in the next major release. Consider looking at Indexing, or the fulltext schema indexes as alternatives.

An index can contain either nodes or relationships.

To create an index with default configuration, simply start using it by adding nodes/relationships to it. It will then be automatically created for you.

What default configuration means depends on how you have configured your database. If you haven't changed any indexing configuration, it means the indexes will be using a Lucene-based backend.

All the examples below show you how to do operations on node indexes, but all of them are just as applicable to relationship indexes. Simply change the "node" part of the URL to "relationship".

If you want to customize the index settings, see Create node index with configuration.

Create node index

Instead of creating the index this way, you can simply start to use it, and it will be created automatically with default configuration.

Example request

- **POST** http://localhost:7474/db/data/index/node/
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "name" : "index_1528894865210_1"
}
```

Example response

- **201:** Created
- **Content-Type:** application/json;charset=utf-8
- **Location:** http://localhost:7474/db/data/index/node/index_1528894865210_1/

```
{
  "template" : "http://localhost:7474/db/data/index/node/index_1528894865210_1/{key}/{value}"
}
```

Create node index with configuration

This request is only necessary if you want to customize the index settings. If you are happy with the defaults, you can just start indexing nodes/relationships, as non-existent indexes will automatically be created as you do. See Configuration and full-text indexing for more information on index configuration.

Example request

- **POST** http://localhost:7474/db/data/index/node/
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "name" : "fulltext",
  "config" : {
    "type" : "fulltext",
    "provider" : "lucene"
  }
}
```

Example response

- **201:** Created
- **Content-Type:** application/json;charset=utf-8
- **Location:** http://localhost:7474/db/data/index/node/fulltext/

```
{
  "template" : "http://localhost:7474/db/data/index/node/fulltext/{key}/{value}",
  "type" : "fulltext",
  "provider" : "lucene"
}
```

Delete node index

Example request

- **DELETE** http://localhost:7474/db/data/index/node/index_1528894864359_1
- **Accept:** application/json; charset=UTF-8

Example response

- **204:** No Content

```

```

List node indexes

Example request

- **GET** http://localhost:7474/db/data/index/node/
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "index_1528894863649_1" : {
    "template" : "http://localhost:7474/db/data/index/node/index_1528894863649_1/{key}/{value}",
    "provider" : "lucene",
    "type" : "exact"
  },
  "index_1528894864229_1" : {
    "template" : "http://localhost:7474/db/data/index/node/index_1528894864229_1/{key}/{value}",
    "provider" : "lucene",
    "type" : "exact"
  },
  "index_1528894863667_1" : {
    "template" : "http://localhost:7474/db/data/index/node/index_1528894863667_1/{key}/{value}",
    "provider" : "lucene",
    "type" : "exact"
  },
  "index_1528894864171_1" : {
    "template" : "http://localhost:7474/db/data/index/node/index_1528894864171_1/{key}/{value}",
    "provider" : "lucene",
    "type" : "exact"
  },
  "index_1528894864538_1" : {
    "template" : "http://localhost:7474/db/data/index/node/index_1528894864538_1/{key}/{value}",
    "provider" : "lucene",
    "type" : "exact"
  },
  "index_1528894863936_1" : {
    "template" : "http://localhost:7474/db/data/index/node/index_1528894863936_1/{key}/{value}",
    "provider" : "lucene",
    "type" : "exact"
  },
  "index_1528894864021_1" : {
    "template" : "http://localhost:7474/db/data/index/node/index_1528894864021_1/{key}/{value}",
    "provider" : "lucene",
    "type" : "exact"
  },
  "index_1528894864093_1" : {
    "template" : "http://localhost:7474/db/data/index/node/index_1528894864093_1/{key}/{value}",
    "provider" : "lucene",
    "type" : "exact"
  },
  "index_1528894864534_1" : {
    "template" : "http://localhost:7474/db/data/index/node/index_1528894864534_1/{key}/{value}",
    "provider" : "lucene",
    "type" : "exact"
  },
  "index_1528894863699_1" : {
    "template" : "http://localhost:7474/db/data/index/node/index_1528894863699_1/{key}/{value}",
    "provider" : "lucene",
    "type" : "exact"
  },
  "index_1528894864012_1" : {
    "template" : "http://localhost:7474/db/data/index/node/index_1528894864012_1/{key}/{value}",
    "provider" : "lucene",
    "type" : "exact"
  },
  "index_1528894864405_1" : {
    "template" : "http://localhost:7474/db/data/index/node/index_1528894864405_1/{key}/{value}",
    "provider" : "lucene",
    "type" : "exact"
  },
  "index_1528894863672_1" : {
    "template" : "http://localhost:7474/db/data/index/node/index_1528894863672_1/{key}/{value}",
    "provider" : "lucene",
    "type" : "exact"
  },
  "index_1528894864459_1" : {
    "template" : "http://localhost:7474/db/data/index/node/index_1528894864459_1/{key}/{value}",
    "provider" : "lucene",
    "type" : "exact"
  }
}
```

Add node to index

Associates a node with the given key/value pair in the given index.

Spaces in the URI have to be encoded as %20.

This does **not** overwrite previous entries. If you index the same key/value/item combination twice, two index entries are created. To do update-type operations, you need to delete the old entry before adding a new one.

Example request

- **POST** http://localhost:7474/db/data/index/node/index_1528894864021_1
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "value" : "some value",
  "uri" : "http://localhost:7474/db/data/node/7",
  "key" : "some-key"
}
```

Example response

- **201:** Created
- **Content-Type:** application/json;charset=utf-8
- **Location:** http://localhost:7474/db/data/index/node/index_1528894864021_1/some-key/some%20value/7

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 7,
    "labels" : [ ]
  },
  "paged_traverse" :
"http://localhost:7474/db/data/node/7/paged/traverse/{returnType}{?pageSize,leaseTime}",
  "outgoing_relationships" : "http://localhost:7474/db/data/node/7/relationships/out",
  "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/7/relationships/out/{-
list|&|types}",
  "create_relationship" : "http://localhost:7474/db/data/node/7/relationships",
  "labels" : "http://localhost:7474/db/data/node/7/labels",
  "traverse" : "http://localhost:7474/db/data/node/7/traverse/{returnType}",
  "all_relationships" : "http://localhost:7474/db/data/node/7/relationships/all",
  "all_typed_relationships" : "http://localhost:7474/db/data/node/7/relationships/all/{-list|&|types}",
  "property" : "http://localhost:7474/db/data/node/7/properties/{key}",
  "self" : "http://localhost:7474/db/data/node/7",
  "incoming_relationships" : "http://localhost:7474/db/data/node/7/relationships/in",
  "properties" : "http://localhost:7474/db/data/node/7/properties",
  "incoming_typed_relationships" : "http://localhost:7474/db/data/node/7/relationships/in/{-
list|&|types}",
  "data" : { },
  "indexed" : "http://localhost:7474/db/data/index/node/index_1528894864021_1/some-key/some%20value/7"
}
```

Remove all entries with a given node from an index

Example request

- **DELETE** http://localhost:7474/db/data/index/node/index_1528894864459_1/12

- **Accept:** application/json; charset=UTF-8

Example response

- **204:** No Content

```
```

Remove all entries with a given node and key from an index

Example request

- **DELETE** http://localhost:7474/db/data/index/node/index_1528894864762_1/kvkey2/15
- **Accept:** application/json; charset=UTF-8

Example response

- **204:** No Content

```
```

Remove all entries with a given node, key and value from an index

Example request

- **DELETE** http://localhost:7474/db/data/index/node/index_1528894864093_1/kvkey1/value1/8
- **Accept:** application/json; charset=UTF-8

Example response

- **204:** No Content

```
```

Find node by exact match

Spaces in the URI have to be encoded as %20.

Example request

- **GET** http://localhost:7474/db/data/index/node/index_1528894865152_1/key/the%2520value
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "metadata" : {
    "id" : 23,
    "labels" : [ ]
  },
  "data" : { },
  "indexed" : "http://localhost:7474/db/data/index/node/index_1528894865152_1/key/the%2520value/23",
  "paged_traverse" :
"http://localhost:7474/db/data/node/23/paged/traverse/{returnType}{?pageSize,leaseTime}",
  "outgoing_relationships" : "http://localhost:7474/db/data/node/23/relationships/out",
  "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/23/relationships/out/{-
list|&|types}",
  "create_relationship" : "http://localhost:7474/db/data/node/23/relationships",
  "labels" : "http://localhost:7474/db/data/node/23/labels",
  "traverse" : "http://localhost:7474/db/data/node/23/traverse/{returnType}",
  "extensions" : { },
  "all_relationships" : "http://localhost:7474/db/data/node/23/relationships/all",
  "all_typed_relationships" : "http://localhost:7474/db/data/node/23/relationships/all/{-list|&|types}",
  "property" : "http://localhost:7474/db/data/node/23/properties/{key}",
  "self" : "http://localhost:7474/db/data/node/23",
  "incoming_relationships" : "http://localhost:7474/db/data/node/23/relationships/in",
  "properties" : "http://localhost:7474/db/data/node/23/properties",
  "incoming_typed_relationships" : "http://localhost:7474/db/data/node/23/relationships/in/{-
list|&|types}"
} ]
```

Find node by query

The query language used here depends on what type of index you are querying. The default index type is Lucene, in which case you should use the Lucene query language here. Below an example of a fuzzy search over multiple keys.

See:
http://lucene.apache.org/core/5_4_0/queryparser/org/apache/lucene/queryparser/classic/package-summary.html

Getting the results with a predefined ordering requires adding the parameter

`order=ordering`

where ordering is one of index, relevance or score. In this case an additional field will be added to each result, named score, that holds the float value that is the score reported by the query result.

Example request

- **GET** http://localhost:7474/db/data/index/node/index_1528894865081_1?query=Name:Build~0.1%20AND%20Gender:Male
- **Accept:** application/json; charset=UTF-8

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
[ {
  "metadata" : {
    "id" : 22,
    "labels" : [ ]
  },
  "data" : {
    "Name" : "Builder"
  },
  "paged_traverse" :
"http://localhost:7474/db/data/node/22/paged/traverse/{returnType}{?pageSize,leaseTime}",
  "outgoing_relationships" : "http://localhost:7474/db/data/node/22/relationships/out",
  "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/22/relationships/out/{-
list|&|types}",
  "create_relationship" : "http://localhost:7474/db/data/node/22/relationships",
  "labels" : "http://localhost:7474/db/data/node/22/labels",
  "traverse" : "http://localhost:7474/db/data/node/22/traverse/{returnType}",
  "extensions" : { },
  "all_relationships" : "http://localhost:7474/db/data/node/22/relationships/all",
  "all_typed_relationships" : "http://localhost:7474/db/data/node/22/relationships/all/{-list|&|types}",
  "property" : "http://localhost:7474/db/data/node/22/properties/{key}",
  "self" : "http://localhost:7474/db/data/node/22",
  "incoming_relationships" : "http://localhost:7474/db/data/node/22/relationships/in",
  "properties" : "http://localhost:7474/db/data/node/22/properties",
  "incoming_typed_relationships" : "http://localhost:7474/db/data/node/22/relationships/in/{-
list|&|types}"
} ]
```

Unique indexing

As of Neo4j 2.0, unique constraints have been added. These make Neo4j enforce the uniqueness, guaranteeing that uniqueness is maintained. See the Getting Started Guide for details about this. For most cases, the unique constraints should be used rather than the features described below.

For uniqueness enforcements, there are two modes:

- URL Parameter `uniqueness=get_or_create`: Create a new node/relationship and index it if no existing one can be found. If an existing node/relationship is found, discard the sent data and return the existing node/relationship.
- URL Parameter `uniqueness=create_or_fail`: Create a new node/relationship if no existing one can be found in the index. If an existing node/relationship is found, return a conflict error.

For details about Neo4j transaction semantics and uniqueness, see the Java Developer Reference.

Get or create unique node (create)

The node is created if it doesn't exist in the unique index already.

Example request

- **POST** http://localhost:7474/db/data/index/node/index_1528370947159_1?uniqueness=get_or_create
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "key" : "name",
  "value" : "Tobias",
  "properties" : {
    "name" : "Tobias",
    "sequence" : 1
  }
}
```

Example response

- **201:** Created
- **Content-Type:** application/json;charset=utf-8
- **Location:** http://localhost:7474/db/data/index/node/index_1528370947159_1/name/Tobias/21

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 21,
    "labels" : [ ]
  },
  "paged_traverse" :
"http://localhost:7474/db/data/node/21/paged/traverse/{returnType}{?pageSize,leaseTime}",
  "outgoing_relationships" : "http://localhost:7474/db/data/node/21/relationships/out",
  "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/21/relationships/out/{-
list|&|types}",
  "create_relationship" : "http://localhost:7474/db/data/node/21/relationships",
  "labels" : "http://localhost:7474/db/data/node/21/labels",
  "traverse" : "http://localhost:7474/db/data/node/21/traverse/{returnType}",
  "all_relationships" : "http://localhost:7474/db/data/node/21/relationships/all",
  "all_typed_relationships" : "http://localhost:7474/db/data/node/21/relationships/all/{-list|&|types}",
  "property" : "http://localhost:7474/db/data/node/21/properties/{key}",
  "self" : "http://localhost:7474/db/data/node/21",
  "incoming_relationships" : "http://localhost:7474/db/data/node/21/relationships/in",
  "properties" : "http://localhost:7474/db/data/node/21/properties",
  "incoming_typed_relationships" : "http://localhost:7474/db/data/node/21/relationships/in/{-
list|&|types}",
  "data" : {
    "sequence" : 1,
    "name" : "Tobias"
  },
  "indexed" : "http://localhost:7474/db/data/index/node/index_1528370947159_1/name/Tobias/21"
}
```

Get or create unique node (existing)

Here, a node is not created but the existing unique node returned, since another node is indexed with the same data already. The node data returned is then that of the already existing node.

Example request

- **POST** http://localhost:7474/db/data/index/node/index_1528370946456_1?uniqueness=get_or_create
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "key" : "name",
  "value" : "Peter",
  "properties" : {
    "name" : "Peter",
    "sequence" : 2
  }
}
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8
- **Location:** http://localhost:7474/db/data/index/node/index_1528370946456_1/name/Peter/11

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 11,
    "labels" : [ ]
  },
  "paged_traverse" :
"http://localhost:7474/db/data/node/11/paged/traverse/{returnType}{?pageSize,leaseTime}",
  "outgoing_relationships" : "http://localhost:7474/db/data/node/11/relationships/out",
  "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/11/relationships/out/{-
list|&|types}",
  "create_relationship" : "http://localhost:7474/db/data/node/11/relationships",
  "labels" : "http://localhost:7474/db/data/node/11/labels",
  "traverse" : "http://localhost:7474/db/data/node/11/traverse/{returnType}",
  "all_relationships" : "http://localhost:7474/db/data/node/11/relationships/all",
  "all_typed_relationships" : "http://localhost:7474/db/data/node/11/relationships/all/{-list|&|types}",
  "property" : "http://localhost:7474/db/data/node/11/properties/{key}",
  "self" : "http://localhost:7474/db/data/node/11",
  "incoming_relationships" : "http://localhost:7474/db/data/node/11/relationships/in",
  "properties" : "http://localhost:7474/db/data/node/11/properties",
  "incoming_typed_relationships" : "http://localhost:7474/db/data/node/11/relationships/in/{-
list|&|types}",
  "data" : {
    "sequence" : 1,
    "name" : "Peter"
  },
  "indexed" : "http://localhost:7474/db/data/index/node/index_1528370946456_1/name/Peter/11"
}
```

Create a unique node or return fail (create)

Here, in case of an already existing node, an error should be returned. In this example, no existing indexed node is found and a new node is created.

Example request

- **POST** http://localhost:7474/db/data/index/node/index_1528370947111_1?uniqueness=create_or_fail
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "key" : "name",
  "value" : "Tobias",
  "properties" : {
    "name" : "Tobias",
    "sequence" : 1
  }
}
```

Example response

- **201:** Created
- **Content-Type:** application/json;charset=utf-8
- **Location:** http://localhost:7474/db/data/index/node/index_1528370947111_1/name/Tobias/20

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 20,
    "labels" : [ ]
  },
  "paged_traverse" :
"http://localhost:7474/db/data/node/20/paged/traverse/{returnType}{?pageSize,leaseTime}",
  "outgoing_relationships" : "http://localhost:7474/db/data/node/20/relationships/out",
  "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/20/relationships/out/{-
list|&|types}",
  "create_relationship" : "http://localhost:7474/db/data/node/20/relationships",
  "labels" : "http://localhost:7474/db/data/node/20/labels",
  "traverse" : "http://localhost:7474/db/data/node/20/traverse/{returnType}",
  "all_relationships" : "http://localhost:7474/db/data/node/20/relationships/all",
  "all_typed_relationships" : "http://localhost:7474/db/data/node/20/relationships/all/{-list|&|types}",
  "property" : "http://localhost:7474/db/data/node/20/properties/{key}",
  "self" : "http://localhost:7474/db/data/node/20",
  "incoming_relationships" : "http://localhost:7474/db/data/node/20/relationships/in",
  "properties" : "http://localhost:7474/db/data/node/20/properties",
  "incoming_typed_relationships" : "http://localhost:7474/db/data/node/20/relationships/in/{-
list|&|types}",
  "data" : {
    "sequence" : 1,
    "name" : "Tobias"
  },
  "indexed" : "http://localhost:7474/db/data/index/node/index_1528370947111_1/name/Tobias/20"
}
```

Create a unique node or return fail (fail)

Here, in case of an already existing node, an error should be returned. In this example, an existing node indexed with the same data is found and an error is returned.

Example request

- **POST** http://localhost:7474/db/data/index/node/index_1528370945887_1?uniqueness=create_or_fail
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "key" : "name",
  "value" : "Peter",
  "properties" : {
    "name" : "Peter",
    "sequence" : 2
  }
}
```

Example response

- **409:** Conflict
- **Content-Type:** application/json;charset=utf-8

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 5,
    "labels" : [ ]
  },
  "paged_traverse" :
"http://localhost:7474/db/data/node/5/paged/traverse/{returnType}{?pageSize,leaseTime}",
  "outgoing_relationships" : "http://localhost:7474/db/data/node/5/relationships/out",
  "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/5/relationships/out/{-
list|&|types}",
  "create_relationship" : "http://localhost:7474/db/data/node/5/relationships",
  "labels" : "http://localhost:7474/db/data/node/5/labels",
  "traverse" : "http://localhost:7474/db/data/node/5/traverse/{returnType}",
  "all_relationships" : "http://localhost:7474/db/data/node/5/relationships/all",
  "all_typed_relationships" : "http://localhost:7474/db/data/node/5/relationships/all/{-list|&|types}",
  "property" : "http://localhost:7474/db/data/node/5/properties/{key}",
  "self" : "http://localhost:7474/db/data/node/5",
  "incoming_relationships" : "http://localhost:7474/db/data/node/5/relationships/in",
  "properties" : "http://localhost:7474/db/data/node/5/properties",
  "incoming_typed_relationships" : "http://localhost:7474/db/data/node/5/relationships/in/{-
list|&|types}",
  "data" : {
    "sequence" : 1,
    "name" : "Peter"
  },
  "indexed" : "http://localhost:7474/db/data/index/node/index_1528370945887_1/name/Peter/5"
}
```

Add an existing node to unique index (not indexed)

Associates a node with the given key/value pair in the given unique index.

In this example, we are using `create_or_fail` uniqueness.

Example request

- **POST** http://localhost:7474/db/data/index/node/index_1528370946955_1?uniqueness=create_or_fail
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "value" : "some value",
  "uri" : "http://localhost:7474/db/data/node/16",
  "key" : "some-key"
}
```

Example response

- **201:** Created
- **Content-Type:** application/json;charset=utf-8
- **Location:** http://localhost:7474/db/data/index/node/index_1528370946955_1/some-key/some%20value/16

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 16,
    "labels" : [ ]
  },
  "paged_traverse" :
"http://localhost:7474/db/data/node/16/paged/traverse/{returnType}{?pageSize,leaseTime}",
  "outgoing_relationships" : "http://localhost:7474/db/data/node/16/relationships/out",
  "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/16/relationships/out/{-
list|&|types}",
  "create_relationship" : "http://localhost:7474/db/data/node/16/relationships",
  "labels" : "http://localhost:7474/db/data/node/16/labels",
  "traverse" : "http://localhost:7474/db/data/node/16/traverse/{returnType}",
  "all_relationships" : "http://localhost:7474/db/data/node/16/relationships/all",
  "all_typed_relationships" : "http://localhost:7474/db/data/node/16/relationships/all/{-list|&|types}",
  "property" : "http://localhost:7474/db/data/node/16/properties/{key}",
  "self" : "http://localhost:7474/db/data/node/16",
  "incoming_relationships" : "http://localhost:7474/db/data/node/16/relationships/in",
  "properties" : "http://localhost:7474/db/data/node/16/properties",
  "incoming_typed_relationships" : "http://localhost:7474/db/data/node/16/relationships/in/{-
list|&|types}",
  "data" : { },
  "indexed" : "http://localhost:7474/db/data/index/node/index_1528370946955_1/some-key/some%20value/16"
}
```

Add an existing node to unique index (already indexed)

In this case, the node already exists in the index, and thus we get a `HTTP 409` status response, as we have set the uniqueness to `create_or_fail`.

Example request

- **POST** http://localhost:7474/db/data/index/node/index_1528370947061_1?uniqueness=create_or_fail
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "value" : "some value",
  "uri" : "http://localhost:7474/db/data/node/19",
  "key" : "some-key"
}
```

Example response

- **409:** Conflict
- **Content-Type:** application/json;charset=utf-8

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 18,
    "labels" : [ ]
  },
  "paged_traverse" :
"http://localhost:7474/db/data/node/18/paged/traverse/{returnType}{?pageSize,leaseTime}",
  "outgoing_relationships" : "http://localhost:7474/db/data/node/18/relationships/out",
  "outgoing_typed_relationships" : "http://localhost:7474/db/data/node/18/relationships/out/{-
list|&|types}",
  "create_relationship" : "http://localhost:7474/db/data/node/18/relationships",
  "labels" : "http://localhost:7474/db/data/node/18/labels",
  "traverse" : "http://localhost:7474/db/data/node/18/traverse/{returnType}",
  "all_relationships" : "http://localhost:7474/db/data/node/18/relationships/all",
  "all_typed_relationships" : "http://localhost:7474/db/data/node/18/relationships/all/{-list|&|types}",
  "property" : "http://localhost:7474/db/data/node/18/properties/{key}",
  "self" : "http://localhost:7474/db/data/node/18",
  "incoming_relationships" : "http://localhost:7474/db/data/node/18/relationships/in",
  "properties" : "http://localhost:7474/db/data/node/18/properties",
  "incoming_typed_relationships" : "http://localhost:7474/db/data/node/18/relationships/in/{-
list|&|types}",
  "data" : {
    "some-key" : "some value"
  },
  "indexed" : "http://localhost:7474/db/data/index/node/index_1528370947061_1/some-key/some%20value/18"
}
```

Get or create unique relationship (create)

Create a unique relationship in an index. If a relationship matching the given key and value already exists in the index, it will be returned. If not, a new relationship will be created.

The type and direction of the relationship is not regarded when determining uniqueness.

Example request

- **POST** http://localhost:7474/db/data/index/relationship/index_1528371006835_1/?uniqueness=get_or_create
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "key" : "name",
  "value" : "Tobias",
  "start" : "http://localhost:7474/db/data/node/27",
  "end" : "http://localhost:7474/db/data/node/28",
  "type" : "knowledge"
}
```

Example response

- **201:** Created
- **Content-Type:** application/json;charset=utf-8
- **Location:** http://localhost:7474/db/data/index/relationship/index_1528371006835_1/name/Tobias/14

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 14,
    "type" : "knowledge"
  },
  "property" : "http://localhost:7474/db/data/relationship/14/properties/{key}",
  "start" : "http://localhost:7474/db/data/node/27",
  "self" : "http://localhost:7474/db/data/relationship/14",
  "end" : "http://localhost:7474/db/data/node/28",
  "type" : "knowledge",
  "properties" : "http://localhost:7474/db/data/relationship/14/properties",
  "data" : {
    "name" : "Tobias"
  },
  "indexed" : "http://localhost:7474/db/data/index/relationship/index_1528371006835_1/name/Tobias/14"
}
```

Get or create unique relationship (existing)

Here, in case of an already existing relationship, the sent data is ignored and the existing relationship returned.

Example request

- **POST** http://localhost:7474/db/data/index/relationship/index_1528371006885_1?uniqueness=get_or_create
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "key" : "name",
  "value" : "Peter",
  "start" : "http://localhost:7474/db/data/node/31",
  "end" : "http://localhost:7474/db/data/node/32",
  "type" : "KNOWS"
}
```

Example response

- **200:** OK
- **Content-Type:** application/json;charset=utf-8

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 15,
    "type" : "KNOWS"
  },
  "property" : "http://localhost:7474/db/data/relationship/15/properties/{key}",
  "start" : "http://localhost:7474/db/data/node/29",
  "self" : "http://localhost:7474/db/data/relationship/15",
  "end" : "http://localhost:7474/db/data/node/30",
  "type" : "KNOWS",
  "properties" : "http://localhost:7474/db/data/relationship/15/properties",
  "data" : { },
  "indexed" : "http://localhost:7474/db/data/index/relationship/index_1528371006885_1/name/Peter/15"
}
```

Create a unique relationship or return fail (create)

Here, in case of an already existing relationship, an error should be returned. In this example, no existing relationship is found and a new relationship is created.

Example request

- **POST** http://localhost:7474/db/data/index/relationship/index_1528371007019_1?uniqueness=create_or_fail
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "key" : "name",
  "value" : "Tobias",
  "start" : "http://localhost:7474/db/data/node/39",
  "end" : "http://localhost:7474/db/data/node/40",
  "type" : "KNOWS"
}
```

Example response

- **201:** Created
- **Content-Type:** application/json;charset=utf-8
- **Location:** http://localhost:7474/db/data/index/relationship/index_1528371007019_1/name/Tobias/18

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 18,
    "type" : "KNOWS"
  },
  "property" : "http://localhost:7474/db/data/relationship/18/properties/{key}",
  "start" : "http://localhost:7474/db/data/node/39",
  "self" : "http://localhost:7474/db/data/relationship/18",
  "end" : "http://localhost:7474/db/data/node/40",
  "type" : "KNOWS",
  "properties" : "http://localhost:7474/db/data/relationship/18/properties",
  "data" : {
    "name" : "Tobias"
  },
  "indexed" : "http://localhost:7474/db/data/index/relationship/index_1528371007019_1/name/Tobias/18"
}
```

Create a unique relationship or return fail (fail)

Here, in case of an already existing relationship, an error should be returned. In this example, an existing relationship is found and an error is returned.

Example request

- **POST** http://localhost:7474/db/data/index/relationship/index_1528371006670_1?uniqueness=create_or_fail
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "key" : "name",
  "value" : "Peter",
  "start" : "http://localhost:7474/db/data/node/19",
  "end" : "http://localhost:7474/db/data/node/20",
  "type" : "KNOWS"
}
```

Example response

- **409:** Conflict
- **Content-Type:** application/json;charset=utf-8

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 10,
    "type" : "KNOWS"
  },
  "property" : "http://localhost:7474/db/data/relationship/10/properties/{key}",
  "start" : "http://localhost:7474/db/data/node/17",
  "self" : "http://localhost:7474/db/data/relationship/10",
  "end" : "http://localhost:7474/db/data/node/18",
  "type" : "KNOWS",
  "properties" : "http://localhost:7474/db/data/relationship/10/properties",
  "data" : { },
  "indexed" : "http://localhost:7474/db/data/index/relationship/index_1528371006670_1/name/Peter/10"
}
```

Add an existing relationship to a unique index (not indexed)

If a relationship matching the given key and value already exists in the index, it will be returned. If not, an `HTTP 409` (conflict) status will be returned in this case, as we are using `create_or_fail`.

It's possible to use `get_or_create` uniqueness as well.

The type and direction of the relationship is not regarded when determining uniqueness.

Example request

- **POST** http://localhost:7474/db/data/index/relationship/index_1528371006614_1?uniqueness=create_or_fail
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "key" : "name",
  "value" : "Peter",
  "uri" : "http://localhost:7474/db/data/relationship/9"
}
```

Example response

- **201:** Created
- **Content-Type:** application/json;charset=utf-8
- **Location:** http://localhost:7474/db/data/index/relationship/index_1528371006614_1/name/Peter/9

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 9,
    "type" : "KNOWS"
  },
  "property" : "http://localhost:7474/db/data/relationship/9/properties/{key}",
  "start" : "http://localhost:7474/db/data/node/15",
  "self" : "http://localhost:7474/db/data/relationship/9",
  "end" : "http://localhost:7474/db/data/node/16",
  "type" : "KNOWS",
  "properties" : "http://localhost:7474/db/data/relationship/9/properties",
  "data" : { },
  "indexed" : "http://localhost:7474/db/data/index/relationship/index_1528371006614_1/name/Peter/9"
}
```

Add an existing relationship to a unique index (already indexed)

Example request

- **POST** http://localhost:7474/db/data/index/relationship/index_1528371006718_1?uniqueness=create_or_fail
- **Accept:** application/json; charset=UTF-8
- **Content-Type:** application/json

```
{
  "key" : "name",
  "value" : "Peter",
  "uri" : "http://localhost:7474/db/data/relationship/12"
}
```

Example response

- **409:** Conflict
- **Content-Type:** application/json;charset=utf-8

```
{
  "extensions" : { },
  "metadata" : {
    "id" : 11,
    "type" : "KNOWS"
  },
  "property" : "http://localhost:7474/db/data/relationship/11/properties/{key}",
  "start" : "http://localhost:7474/db/data/node/21",
  "self" : "http://localhost:7474/db/data/relationship/11",
  "end" : "http://localhost:7474/db/data/node/22",
  "type" : "KNOWS",
  "properties" : "http://localhost:7474/db/data/relationship/11/properties",
  "data" : { },
  "indexed" : "http://localhost:7474/db/data/index/relationship/index_1528371006718_1/name/Peter/11"
}
```

WADL Support

The Neo4j REST API is a truly RESTful interface relying on hypermedia controls (links) to advertise permissible actions to users. Hypermedia is a dynamic interface style where declarative constructs (semantic markup) are used to inform clients of their next legal choices just in time.

RESTful APIs cannot be modeled by static interface description languages like WSDL or WADL.

However for some use cases, developers may wish to expose WADL descriptions of the Neo4j REST API, particularly when using tooling that expects such.

In those cases WADL generation may be enabled by adding to *neo4j.conf*:

```
unsupported.dbms.wadl_generation_enabled=true
```

WADL is not an officially supported part of the Neo4j server API because WADL is insufficiently expressive to capture the set of potential interactions a client can drive with Neo4j server. Expect the WADL description to be incomplete, and in some cases contradictory to the real API. In any cases where the WADL description disagrees with the REST API, the REST API should be considered authoritative. WADL generation may be withdrawn at any point in the Neo4j release cycle.

Using the REST API from WebLogic

When deploying an application to WebLogic you may run into problems when Neo4j responds with an HTTP status of `204 No Content`. The response does not contain an entity body in such cases.

This can cause WebLogic to throw `java.net.SocketTimeoutException: Read timed out` for no obvious reason.

If you encounter this, please try setting `UseSunHttpHandler` to `true`. You can for example do this by adding the following to the WebLogic startup script:

```
-DUseSunHttpHandler=true
```

The WebLogic startup script is called `bin\startWebLogic.sh` (`bin/startWebLogic.cmd` on Windows).

License