### <https://neo4j.com/docs/rest-docs/current/>

<http://neo4j.com/docs/developer-manual/3.0/http-api/#http-api-transactional>

<http://neo4j.com/docs/developer-manual/3.0/cypher/>

### 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://46.101.180.63:7474/db/data/
* Accept: application/json; charset=UTF-8

*Example response*

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

{

"extensions": {

"SPARQLPlugin": {

"execute\_sparql": "http://46.101.180.63:7474/db/data/ext/SPARQLPlugin/graphdb/execute\_sparql",

"insert\_quad": "http://46.101.180.63:7474/db/data/ext/SPARQLPlugin/graphdb/insert\_quad"

}

},

"node": "http://46.101.180.63:7474/db/data/node",

"relationship": "http://46.101.180.63:7474/db/data/relationship",

"node\_index": "http://46.101.180.63:7474/db/data/index/node",

"relationship\_index": "http://46.101.180.63:7474/db/data/index/relationship",

"extensions\_info": "http://46.101.180.63:7474/db/data/ext",

"relationship\_types": "http://46.101.180.63:7474/db/data/relationship/types",

"batch": "http://46.101.180.63:7474/db/data/batch",

"cypher": "http://46.101.180.63:7474/db/data/cypher",

"indexes": "http://46.101.180.63:7474/db/data/schema/index",

"constraints": "http://46.101.180.63:7474/db/data/schema/constraint",

"transaction": "http://46.101.180.63:7474/db/data/transaction",

"node\_labels": "http://46.101.180.63:7474/db/data/labels",

"neo4j\_version": "3.0.4"  
}

### 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" ] ]  
}

### Send a query

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

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 ] ]  
}

## 

### 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

[ "name", "foo" ]

***Example request get transaction***

* POST http://46.101.180.63:7474/db/data/transaction/commit
* Accept: application/json; charset=UTF-8
* Content-Type: application/json

{

"statements" : [ {

"statement" : "MATCH (n:BTC\_Address)-[r:Inputs\_to]->(m:Transaction) MATCH (m:Transaction)-[x:Outputs\_to]->(p:BTC\_Address) where p<>n AND m.hsh='2e9f40d4af66964a526d51233f87c401eaf66e05afd1e65928d2f2a51e636232' Optional MATCH (a:Person)-[h:HAS]->(p) RETURN n,r,m,x,p,a LIMIT 50"

} ]

}

*Example response*

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

{

"results": [

{

"columns": [

"n",

"r",

"m",

"x",

"p",

"a"

],

"data": [

{

"row": [

{

"unconfirmedBalance": 0,

"address": "1LuckyG4tMMZf64j6ea7JhCz7sDpk6vdcS",

"balance": 0.02016505,

"totalSent": 38141.55498487,

"totalReceived": 38141.57514992

},

{

"sequence": 4294967295,

"scriptSig": "30450221009a192cb7918eeccdae9e20783c029588dc601477d34645ee1b1b57453383789202200679fe452d500e2dc2c61ad64d22a900c60eefdc70d6ca75525d4c5fa9470f96[ALL] 04da6bc6a6139bb008454bfc8371141a5fb8ba6de87e9ab1578ab4c31e1b25513d6d1b1b0e66b0e39a29f6baf19f9f0faaf51d22bac02b1c07eb08058498763784",

"value": 0.01113174

},

{

"fees": 0.0001,

"size": 224,

"locktime": 0,

"valueIn": 0.01113174,

"hsh": "2e9f40d4af66964a526d51233f87c401eaf66e05afd1e65928d2f2a51e636232",

"confirmations": 72009,

"version": 1,

"valueOut": 0.01103174

},

{

"scriptPubKey": "OP\_DUP OP\_HASH160 e23120c087600dbc56cda24cdb545b800a993a3d OP\_EQUALVERIFY OP\_CHECKSIG",

"type": "pubkeyhash",

"value": "0.01103174"

},

{

"unconfirmedBalance": 0,

"address": "1MczZpiLS1gYC2buMUsDKCxd1yTJFD4TDh",

"balance": 0.00096363,

"totalSent": 32.90252539,

"totalReceived": 32.90348902

},

{

"PostalAddress": "",

"name": "Christian"

}

],

"meta": [

{

"id": 330,

"type": "node",

"deleted": false

},

{

"id": 53,

"type": "relationship",

"deleted": false

},

{

"id": 332,

"type": "node",

"deleted": false

},

{

"id": 54,

"type": "relationship",

"deleted": false

},

{

"id": 320,

"type": "node",

"deleted": false

},

{

"id": 319,

"type": "node",

"deleted": false

}

]

}

]

}

],

"errors": []

}