RedisJSON quick start

Prerequisites

For this quick start tutorial, you need:

- A Redis database with the RedisJSON module enabled. You can use either:
  - A Redis Cloud database
  - A Redis Enterprise Software database
- redis-cli command line tool
- redis-py client library v4.0.0 or greater

RedisJSON with redis-cli

The redis-cli command-line tool is part of the Redis installation. You can use it to connect to your Redis database and test RedisJSON commands.

In these examples, you will create a shopping list using a JSON document in Redis.

Connect to a database

```
$ redis-cli -h <endpoint> -p <port> -a <password> --raw
127.0.0.1:12543>
```

The --raw option maintains the format of the database response.

Create a JSON document

Use the JSON.SET command to set a Redis key with a JSON value. A key can contain any valid JSON value, including scalars, objects, and arrays.

Set the key shopping-list to a simple JSON object containing the date the list was created.

```
127.0.0.1:12543> JSON.SET shopping-list $ '{"list-date": "05/05/2022"}'
OK
```

The second argument in JSON.SET is the path in the JSON object. The $ or . character is the “root” of the JSON object. All new keys must have either $ or . as the path.

Add info to a JSON document

Now that you have a JSON object, use JSON.SET to add an entity to it. You can add any JSON entity type to a JSON object with JSON.SET.

In the shopping-list JSON key, use JSON.SET to add an object with the path .stores. Use JSON.SET to add other objects to the .stores path to represent the stores you need to visit.
In each store, add an array of items that will represent the number of specific items you need to buy from those stores.

```
127.0.0.1:12543> JSON.SET shopping-list .stores .grocery-store '{ "items": [ { "name": "apples", "count": 5 } ] }'
OK
```

```
127.0.0.1:12543> JSON.SET shopping-list .stores .hardware-store '{ "items": [ { "name": "hammers", "count": 1 } ] }'
OK
```

```
127.0.0.1:12543> JSON.SET shopping-list .stores .clothing-store '{ "items": [ { "name": "socks", "count": 2 } ] }'
OK
```

Use **JSON.ARRAPPEND** to add more items to the items arrays. **JSON.ARRAPPEND** returns the number of objects in the array.

```
127.0.0.1:12543> JSON.ARRAPPEND shopping-list .stores .grocery-store .items '{ "name": "pears", "count": 3 }'
2
```

Use **JSON.NUMINCRBY** to change the number of a specific item that you need. This returns the new value of the item.

```
127.0.0.1:12543> JSON.NUMINCRBY shopping-list .stores .clothing-store .items[0].count 2
4
127.0.0.1:12543> JSON.NUMINCRBY shopping-list .stores .grocery-store .items[1] count -1
2
```

**Read a JSON document**

Use **JSON.GET** to read the JSON object from the database. If you connected to `redis-cli` with the **--raw** option, you can format the response to **JSON.GET** with the **INDENT**, **NEWLINE**, and **SPACE** options.
You can also use JSON.GET to read a single entity or multiple entities with the same name from the JSON object.
Use JSON.TYPE to check the JSON type of the key or an entity inside the key.

Use JSON.DEL to delete parts of the JSON document.
Use JSON.DEL with no specified path to delete the key.

RedisJSON with Python

If you want to use RedisJSON within an application, you can use one of the client libraries.

The following example uses the Redis Python client library redis-py, which supports RedisJSON commands as of v4.0.0.

This Python code creates a JSON document in Redis, adds and updates information to the JSON document, and then deletes the document.

```python
import redis
import json

# Connect to a database
r = redis.Redis(host="<endpoint>", port="<port>",
                password="<password>")

# Create a JSON document
print("Creating shopping list...")
list_obj = {
    'list-date': '05/05/2022'
}
```
r.json().set('shopping-list:py', '.', list_obj)
reply = r.json().get('shopping-list:py', '.
print(json.dumps(reply, indent=4) + "\n"

# Add info to the JSON document
print("Adding stores and starting items...")
stores_obj = {
    "grocery-store" : {
        "items" : [ { "name": "apples", "count": 5 } ]
    },
    "hardware-store" : {
        "items": [ { "name": "hammers", "count": 1 } ]
    },
    "clothing-store" : {
        "items": [ { "name": "socks", "count": 2 } ]
    }
}

r.json().set('shopping-list:py', '.stores', stores_obj)
reply = r.json().get('shopping-list:py', '.
print(json.dumps(reply, indent=4) + "\n"

# Add new items to the list
print("Adding pears...")
pears_obj = {
    "name" : "pears",
    "count" : 3
}

r.json().arrappend('shopping-list:py', '.stores.grocery-store.items',
    pears_obj)
reply = r.json().get('shopping-list:py', '.
print(json.dumps(reply, indent=4) + "\n"

# Increment item counts
print("Changing item counts...")
r.json().numincrby('shopping-list:py',
    '.stores.clothing-store.items[0].count', 2)
r.json().numincrby('shopping-list:py',
    '.stores.grocery-store.items[1].count', -1)
reply = r.json().get('shopping-list:py', '.
print(json.dumps(reply, indent=4) + "\n"

# Get all items no matter which heading they're under
print("Getting all items...")
reply = r.json().get('shopping-list:py', '$..items')
print(json.dumps(reply, indent=4) + "\n"

# Delete specific parts of the document
print("Deleting clothing store and pears...")
r.json().delete('shopping-list:py', '.stores.clothing-store')
r.json().delete('shopping-list:py', '.stores.grocery-store.items[1]')
reply = r.json().get('shopping-list:py', '.
print(json.dumps(reply, indent=4) + "\n"

# Delete the JSON document key

print("Deleting shopping-list:py key...")
r.json().delete('shopping-list:py')
print("Done!")
Changing item counts...
{
   "list-date": "05/05/2022",
   "stores": {
      "grocery-store": {
         "items": [
            {
               "name": "apples",
               "count": 5
            },
            {
               "name": "pears",
               "count": 2
            }
         ]
      },
      "hardware-store": {
         "items": [
            {
               "name": "hammers",
               "count": 1
            }
         ]
      },
      "clothing-store": {
         "items": [
            {
               "name": "socks",
               "count": 4
            }
         ]
      }
   }
}
Getting all items...
[
  [
    {
      "name": "apples",
      "count": 5
    },
    {
      "name": "pears",
      "count": 2
    }
  ],
  [
    {
      "name": "hammers",
      "count": 1
    }
  ],
  [
    {
      "name": "socks",
      "count": 4
    }
  ]
]

Deleting clothing store and pears...
{
  "list-date": "05/05/2022",
  "stores": {
    "grocery-store": {
      "items": [
        {
          "name": "apples",
          "count": 5
        }
      ]
    },
    "hardware-store": {
      "items": [
        {
          "name": "hammers",
          "count": 1
        }
      ]
    }
  }
}

Deleting shopping-list:py key...
Done!

More info
- RedisJSON commands
RedisJSON client libraries

Updated: May 13, 2022