Support Portal > Requests

How up Setup Redis to use RESP3

Ticket details

Jose Perez Saborio
21 hours ago

Hi Alberto,

Glad the deep dive sparked more questions—let me clarify:

1. What `my_hgetall` actually returns
When you register a Redis Function and call `redis.setresp(3)`, subsequent `redis.call('HGETALL', hash)` returns a RESP3 **dictionary** reply. In Lua, you see this as a special table whose associative part lives under `res.map`:

```
redis.setresp(3)
local res = redis.call('HGETALL', hash)
-- res.map is a Lua table like { field1 = "value1", field2 = "value2", … }
res.map["_last_modified_"] = nil   -- remove the metadata return res
```

When you run:

```
redis-cli -3 FCALL my_hgetall 1 myhash
```

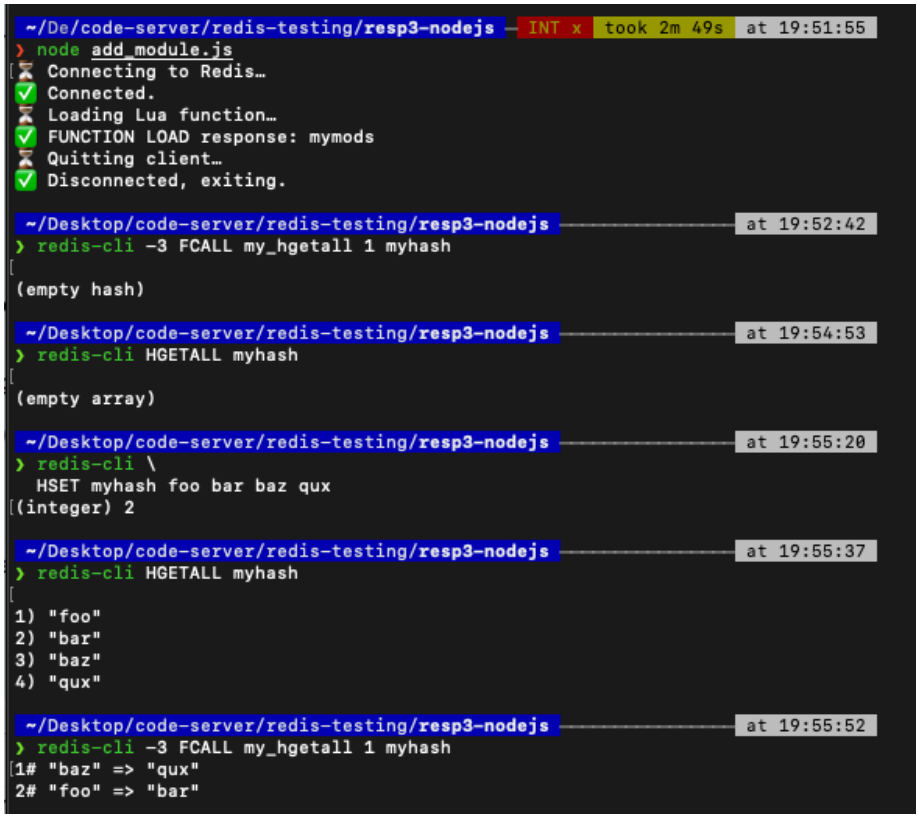
You'll get just the remaining field/value pairs (no `_last_modified_`), see screenshot at the end.

2. Why use RESP3 rather than JSON-encode every time?
RESP3's built-in **map** type is perfect for flat key→value results—clients automatically deserialize it into objects or dictionaries without extra work. You only need to switch to `cjson.encode()` if you want to return more complex or nested Lua tables that RESP3 doesn't natively support (for example, a list of maps or deeply nested structures). For a simple filtered hash like `my_hgetall`, RESP3 alone is enough.

3. Why not just do `res["_last_modified_"] = nil`?
In this Functions API, the hash fields live in `res.map`, not directly on `res`. Setting `res["_last_modified_"]`

`= nil` won't remove anything because `res` itself has no such key, so you must nil out the metadata inside the `map` subtable.

Hope this helps! Let me know if you'd like any more examples or have other questions. I'm attaching a few samples that you can use for testing purposes. Have a great weekend!



Best,
Jose
Customer Success
Redis

my_hgetall.lua
255 Bytes · Download

add_module.js
872 Bytes · Download

Alberto Iong
2 days ago

Hi Jose,

Thanks for your in depth explanation... My story began here, while peeping into [Redis Function](https://redis.io/docs/latest/develop/programmability/functions-intro/) documentation, the code example
```
local function my_hgetall(keys, args)
```

```
 redis.setresp(3)
 local hash = keys[1]
 local res = redis.call('HGETALL', hash)
 res['map']['_last_modified_'] = nil
 return res
end
```

> While all of the above should be straightforward, note that the my_hgetall also calls redis.setresp(3). That means that the function expects RESP3 replies after calling redis.call(), which, unlike the default RESP2 protocol, provides dictionary (associative arrays) replies. Doing so allows the function to delete (or set to nil as is the case with Lua tables) specific fields from the reply, and in our case, the _last_modified_ field.

arousd me upmost curiosity... And according to your reply, the use of "redis.setresp(3)" doesn't make "my_hgetall" return an object automatically. To be honest, I didn't test this code yet, in this case what does "my_hgetall" returns?

And if it is necessary to use "cjson.encode()" to serialize the returned object. what's point in using RESP3? It seems to me that the sole purpose is to remove the '_last_modified_' field... and if it is so why not use this?

```
res['_last_modified_'] = nil
```

Thanks for your reply and have a nice weekend.

Alberto

Jose Perez Saborio
3 days ago

Hi Alberto,

I appreciate your patience.

I want to clarify this question we often see when working with Redis RESP3 and Lua scripting, specifically for your scenario. I'll explain why certain operations return an empty array [] when accessed from JavaScript or via redis-cli and what steps you can take if your goal is to get structured, non-empty data back.

Starting from Redis 6.0, RESP3 became the new default write protocol, and it brought more explicit distinction between "empty" collections and "null" responses:

- **Empty collections** (like a list, set, or map with zero items) are returned as an empty array (`*0\r\n` in raw RESP3, displayed as `[]` in clients).
- **Nulls/absence of value** (e.g., a missing hash field or nonexistent key) are returned as RESP3 null (`_\r\n`).

This change impacts how redis-cli and modern JavaScript/Node.js clients display and interpret results.

Redis Lua scripting supports returning Lua tables. The way a Lua table is returned depends on its structure:

- **Numerically-indexed tables (arrays like `{1, 2, 3}`)** are correctly serialized to RESP3 arrays. These display as a non-empty array in both JavaScript and redis-cli.
- **Tables with string keys (e.g. `{ name = "iong_dev", status = "active" }`)** are "object-like" from Lua's perspective, but Redis can't serialize these directly as RESP3 maps in EVAL. Instead:
  - Redis returns an empty array (`[]`) under RESP3.
  - redis-cli shows `(empty array)` .

This response is by design, not an error.

In your screenshot, you're doing something similar to this:

```
redis-cli -3 \
  -h host \
  -p 6379 \
  EVAL "redis.setresp(3); return { name = 'iong_dev', status = 'active', score = 98 }" 0
```

Returns:

```
(empty array)
```

This is expected because the returned table is not numerically indexed.

If your goal is to receive an actual map/object (not an empty array):

Solution: Encode as JSON in Lua

Instead of returning a Lua table directly, encode it into a JSON string within your Lua script. For example:

```
redis.setresp(3)
return cjson.encode({ name = "iong_dev", status = "active", score = 98 })
```

- **In JavaScript**, parse this string using `JSON.parse()` to get a native object:

```
const hello = await client.sendCommand(['HELLO', '3']);
console.log('HELLO reply:', hello);

const lua = `
    redis.setresp(3);
    return cjson.encode({ name = "iong_dev", status = "active", score = 98 });
  `;
const result = await client.sendCommand([
    'EVAL',
    lua,
    '0'
]);


console.log('Raw EVAL result:', result); // JSON string

const data = JSON.parse(result);
console.log('Parsed object:', data);
```

```
❯ node testResp3.js
HELLO reply: [Object: null prototype] {
  server: 'redis',
  version: '7.4.3',
  proto: 3,
  id: 117,
  mode: 'cluster',
  role: 'master',
  modules: [
    [Object: null prototype] {
      name: 'bf',
      ver: 20807,
      path: '/enterprise-managed',
      args: []
    },
    [Object: null prototype] {
      name: 'timeseries',
      ver: 11207,
      path: '/enterprise-managed',
      args: []
    },
    [Object: null prototype] {
      name: 'searchlight',
      ver: 21020,
      path: '/enterprise-managed',
      args: []
    },
    [Object: null prototype] {
      name: 'ReJSON',
      ver: 20811,
      path: '/enterprise-managed',
      args: []
    }
  ]
}
Raw EVAL result: {"status":"active","name":"iong_dev","score":98}
Parsed object: { status: 'active', name: 'iong_dev', score: 98 }
Disconnected from Redis
```

- **In redis-cli**, you'll see the raw JSON string, which you can read or process as needed.

```
❯ redis-cli -3
127.0.0.1:6379> HELLO 3
1# "server" => "redis"
2# "version" => "8.0.2"
3# "proto" => (integer) 3
4# "id" => (integer) 10
5# "mode" => "standalone"
6# "role" => "master"
7# "modules" =>
```

```
❯ docker run --rm -it redis:latest redis-cli -3 \
  -h host.docker.internal -p 6379 \
  EVAL "redis.setresp(3); return cjson.encode({ name = 'iong_dev', status = 'active', score = 98 })" \
  0
"{\"status\":\"active\",\"name\":\"iong_dev\",\"score\":98}"
```

With RESP3 and JavaScript clients, an empty array ([]) on return—especially from { key = val, ... } Lua tables—is expected. To reliably send richer objects from Lua to JavaScript, always JSON-encode your return values and parse the result.

Feel free to let me know if you need additional examples or additional details on this topic!

Best regards,
Jose
Customer Success
Redis

Jose Perez Saborio
3 days ago

Hi Alberto,

Thank you for sharing the details and code samples regarding your attempts to use RESP3 with Redis Cloud and your local Redis 8 instance. I'm investigating this issue further to provide you with the most accurate guidance. I appreciate your patience and will update you as soon as I have more information.

If you have any additional logs or error details, please share them—they may help expedite the troubleshooting process.

Best regards,
Jose
Customer Success
Redis

Alberto Iong
4 days ago

Hi,

Just to add one more image using redis-cli, the protocol is confirmed as RESP3, but the return object is forced back to RESP2... I think...

resp3test-2.JPG
70 KB · Download

Alberto Iong
5 days ago

Hi,

This is the nodejs program i wrote to test resp3 with:

```
import { redis } from './redis/redis.js'

/*
  main
*/

const script = `
   redis.setresp(3)
   return { name = "iong_dev", status = 'active', score = 98 }
 `

await redis.connect()

await redis.sendCommand(['HELLO', '3']);

console.log(await redis.sendCommand(['HELLO']));

console.log('The result is', await redis.eval(script, 0))

await redis.close();

process.exit(0)
```

and the output is attached. I test with my Redis Cloud account and local instance on Redis 8.

resp3test.JPG
60 KB · Download

Jose Perez Saborio
5 days ago

Hi Alberto,

My name is Jose from the Redis Customer Success team—thanks for reaching out!

Please let me know whether you use Redis Cloud or self-managed Redis Software to help you better. Additionally, could you share more details about your issues when setting up RESP3? Any error messages, stack traces, or logs from your application would be beneficial. This information will allow us to provide more targeted guidance for your use case.

I'm looking forward to hearing back from you!

Best regards,
Jose
Customer Success
Redis

Alberto Iong
5 days ago

This is the nodejs program i wrote to test resp3 with:

```
import { redis } from './redis/redis.js'

/*
  main
*/

const script = `
   redis.setresp(3)
   return { name = "iong_dev", status = 'active', score = 98 }
 `

await redis.connect()

await redis.sendCommand(['HELLO', '3']);

console.log(await redis.sendCommand(['HELLO']));

console.log('The result is', await redis.eval(script, 0))

await redis.close();
```

process.exit(0)

```

and the output is attached.
resp3test.JPG
60 KB · Download

---

Parag Marjiwe
5 days ago

Hi Alberto,

Thank you for reaching out to Redis support regarding queries related to setting up redis to use RESP3.

We have forwarded your request to the relevant team, who will get back to you during local business hours. Meanwhile, we appreciate your patience and understanding.

Best Regards,
Parag
The Redis Team
Blog | X | LinkedIn

---

Redis Support Bot Agent
5 days ago

Hello,

This is an automated response.
One of our Support Engineers will get back to you soon.

Note that, based on our policies, we will never ask for your passwords or credentials.

Regards,
Redis Support Automated Services

University ↗ Blog ↗

---

Alberto Iong
5 days ago
```

Hi hi,

I was trying to setup Redis to response in RESP3 in Redis Function and Lua Script but failed. Is there any technical guide to teach how to setup RESP3 ? The drivers setup, redis.conf … etc

Alberto

Add to conversation

Mark as solved

Support Portal