

---

## Parse Redis dump.rdb files, Analyze Memory, and Export Data to JSON

Rdbtools is a parser for Redis' dump.rdb files. The parser generates events similar to an xml sax parser, and is very efficient memory wise.

In addition, rdbtools provides utilities to :

1. Generate a Memory Report of your data across all databases and keys
2. Convert dump files to JSON
3. Compare two dump files using standard diff tools

Rdbtools is written in Python, though there are similar projects in other languages. See FAQs for more information.

See <https://rdbtools.com> for a gui to administer redis, commercial support, and other enterprise features.

### Installing rdbtools

Pre-Requisites :

1. python-lzf is optional but highly recommended to speed up parsing.
2. redis-py is optional and only needed to run test cases.

To install from PyPI (recommended) :

```
1 pip install rdbtools python-lzf
```

To install from source :

```
1 git clone https://github.com/sripathikrishnan/redis-rdb-tools
2 cd redis-rdb-tools
3 sudo python setup.py install
```

### Command line usage examples

Every run of RDB Tool requires to specify a command to indicate what should be done with the parsed RDB data. Valid commands are: json, diff, justkeys, justkeyvals and protocol.

JSON from a two database dump:

```
1 > rdb --command json /var/redis/6379/dump.rdb
2
```

---

```
3 [{
4   "user003":{"fname":"Ron","sname":"Bumquist"},
5   "lizards":["Bush anole","Jackson's chameleon","Komodo dragon","Ground
      agama","Bearded dragon"],
6   "user001":{"fname":"Raoul","sname":"Duke"},
7   "user002":{"fname":"Gonzo","sname":"Dr"},
8   "user_list":["user003","user002","user001"]}, {
9   "balloon":{"helium":"birthdays","medical":"angioplasty","weather":"
      meteorology"},
10  "armadillo":["chacoan naked-tailed","giant","Andean hairy","nine-banded
      ","pink fairy"],
11  "aroma":{"pungent":"vinegar","putrid":"rotten eggs","floral":"roses"}}}]
```

## Filter parsed output

Only process keys that match the regex, and only print key and values:

```
1 > rdb --command justkeyvals --key "user.*" /var/redis/6379/dump.rdb
2
3 user003 fname Ron,sname Bumquist,
4 user001 fname Raoul,sname Duke,
5 user002 fname Gonzo,sname Dr,
6 user_list user003,user002,user001
```

Only process hashes starting with “a”, in database 2:

```
1 > rdb -c json --db 2 --type hash --key "a.*" /var/redis/6379/dump.rdb
2
3 [{},{
4   "aroma":{"pungent":"vinegar","putrid":"rotten eggs","floral":"roses"}}}]
```

## Converting dump files to JSON

The `json` command output is UTF-8 encoded JSON. By default, the callback try to parse RDB data using UTF-8 and escape non ‘ASCII printable’ characters with the `\u` notation, or non UTF-8 parsable bytes with `\x`. Attempting to decode RDB data can lead to binary data corruption, this can be avoided by using the `--escape raw` option. Another option, is to use `-e base64` for Base64 encoding of binary data.

Parse the dump file and print the JSON on standard output:

```
1 > rdb -c json /var/redis/6379/dump.rdb
2
3 [{
4   "Citat":["B\u00e4ttre sent \u00e4n aldrig","Bra karl reder sig sj\
      u\u00e4lv","Man ska inte k\u00f6pa grisen i s\u00e4cken"],
```

---

```
5 "bin_data": "\\xFE\\u0000\\u00e2\\xF2"]}]
```

Parse the dump file to raw bytes and print the JSON on standard output:

```
1 > rdb -c json /var/redis/6379/dump.rdb --escape raw
2
3 [{
4 "Citat":["B\u00c3\u00a4ttre sent \u00c3\u00a4n aldrig","Bra karl reder
    sig sj\u00c3\u00a4lv","Man ska inte k\u00c3\u00b6pa grisen i s\u00c3
    \u00a4cken"],
5 "bin_data": "\u00fe\u0000\u00c3\u00a2\u00f2"]}]
```

## Generate Memory Report

Running with the `-c memory` generates a CSV report with the approximate memory used by that key. `--bytes C` and `'--largest N` can be used to limit output to keys larger than C bytes, or the N largest keys.

```
1 > rdb -c memory /var/redis/6379/dump.rdb --bytes 128 -f memory.csv
2 > cat memory.csv
3
4 database,type,key,size_in_bytes,encoding,num_elements,
    len_largest_element
5 0,list,lizards,241,quicklist,5,19
6 0,list,user_list,190,quicklist,3,7
7 2,hash,baloon,138,ziplist,3,11
8 2,list,armadillo,231,quicklist,5,20
9 2,hash,aroma,129,ziplist,3,11
```

The generated CSV has the following columns - Database Number, Data Type, Key, Memory Used in bytes and RDB Encoding type. Memory usage includes the key, the value and any other overheads.

Note that the memory usage is approximate. In general, the actual memory used will be slightly higher than what is reported.

You can filter the report on keys or database number or data type.

The memory report should help you detect memory leaks caused by your application logic. It will also help you optimize Redis memory usage.

## Find Memory used by a Single Key

Sometimes you just want to find the memory used by a particular key, and running the entire memory report on the dump file is time consuming.

In such cases, you can use the `redis-memory-for-key` command:

---

```
1 > redis-memory-for-key person:1
2
3 > redis-memory-for-key -s localhost -p 6379 -a mypassword person:1
4
5 Key           person:1
6 Bytes         111
7 Type          hash
8 Encoding      ziplist
9 Number of Elements  2
10 Length of Largest Element  8
```

NOTE :

1. This was added to redis-rdb-tools version 0.1.3
2. This command depends redis-py package

## Comparing RDB files

First, use the `--command diff` option, and pipe the output to standard sort utility

```
1 > rdb --command diff /var/redis/6379/dump1.rdb | sort > dump1.txt
2 > rdb --command diff /var/redis/6379/dump2.rdb | sort > dump2.txt
```

Then, run your favourite diff program

```
1 > kdiff3 dump1.txt dump2.txt
```

To limit the size of the files, you can filter on keys using the `--key` option

## Emitting Redis Protocol

You can convert RDB file into a stream of redis protocol using the `protocol` command.

```
1 > rdb -c protocol /var/redis/6379/dump.rdb
2
3 *4
4 $4
5 HSET
6 $9
7 users:123
8 $9
9 firstname
10 $8
11 Sripathi
```

---

You can pipe the output to netcat and re-import a subset of the data. For example, if you want to shard your data into two redis instances, you can use the `-key` flag to select a subset of data, and then pipe the output to a running redis instance to load that data. Read Redis Mass Insert for more information on this.

When printing protocol output, the `--escape` option can be used with `printable` or `utf8` to avoid non printable/control characters.

By default, expire times are emitted verbatim if they are present in the rdb file, causing all keys that expire in the past to be removed. If this behaviour is unwanted the `-x/--no-expire` option will ignore all key expiry commands.

Otherwise you may want to set an expiry time in the future with `-a/--amend-expire` option which adds an integer number of seconds to the expiry time of each key which is already set to expire. This will not change keys that do not already have an expiry set.

## Using the Parser

```
1 from rdbtools import RdbParser, RdbCallback
2 from rdbtools.encodehelpers import bytes_to_unicode
3
4 class MyCallback(RdbCallback):
5     ''' Simple example to show how callback works.
6         See RdbCallback for all available callback methods.
7         See JsonCallback for a concrete example
8     '''
9
10    def __init__(self):
11        super(MyCallback, self).__init__(string_escape=None)
12
13    def encode_key(self, key):
14        return bytes_to_unicode(key, self._escape, skip_printable=True)
15
16    def encode_value(self, val):
17        return bytes_to_unicode(val, self._escape)
18
19    def set(self, key, value, expiry, info):
20        print('%s = %s' % (self.encode_key(key), self.encode_value(
21            value)))
22
23    def hset(self, key, field, value):
24        print('%s.%s = %s' % (self.encode_key(key), self.encode_key(
25            field), self.encode_value(value)))
26
27    def sadd(self, key, member):
```

---

```
26         print('%s has {%s}' % (self.encode_key(key), self.encode_value(
27             member)))
28     def rpush(self, key, value):
29         print('%s has [%s]' % (self.encode_key(key), self.encode_value(
30             value)))
31     def zadd(self, key, score, member):
32         print('%s has {%s : %s}' % (str(key), str(member), str(score)))
33
34
35 callback = MyCallback()
36 parser = RdbParser(callback)
37 parser.parse('/var/redis/6379/dump.rdb')
```

## Other Pages

1. Frequently Asked Questions
2. Redis Dump File Specification
3. Redis Dump File Version History - this also has notes on converting a dump file to an older version.

## License

rdbtools is licensed under the MIT License. See LICENSE

## Maintained By

Sripathi Krishnan : @srithedabbler

## Credits

1. Didier Spézia
2. Yoav Steinberg
3. Daniel Mezzatto
4. Carlo Cabanilla
5. Josep M. Pujol
6. Charles Gordon
7. Justin Poliey