**Title:** Redis BRPOP

**Excerpt:** Redis list is a collection of string values stored at a given key. Among several commands, the BRPOP command is used to pop elements from the tail of the list while supporting the blocking behavior where the client will wait for a non-empty list until the timeouts.

**Permalink:** redis-brpop

**Category:** Redis

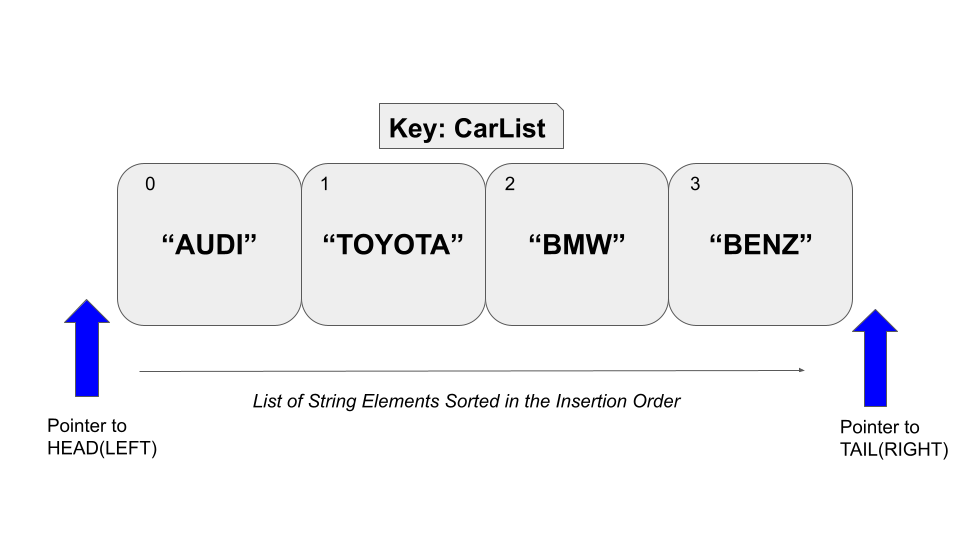
# 

# 

# 

# Introduction to Redis Lists

The list is a popular Redis data type that stores a sequence of strings based on the insertion order. One list can hold more than 4 billion elements. The unique fact about the Redis list is it maintains HEAD and TAIL properties where the elements can be inserted from both sides. Usually, the *LPUSH* and *RPUSH* commands are used to insert elements into a list. Meantime, the brand new list will be created with the specified key. All the list commands behave the same way where a new list will be created when passed with a non-existing key.

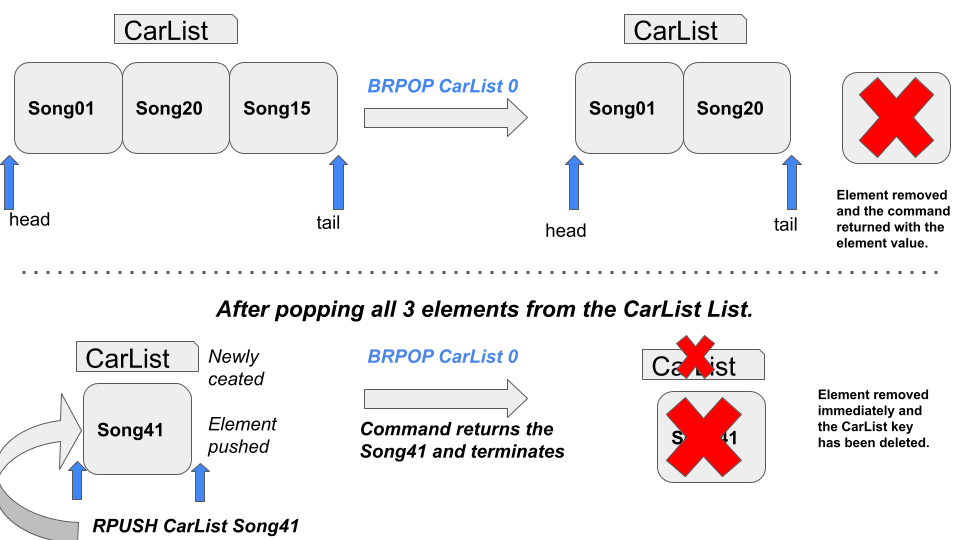


In addition, when all the elements are popped from the list, the associated key will be deleted from the Redis key space. Redis list is the ideal candidate for low latency applications because the insertion and deletion operations have constant time complexity at the *HEAD* and *TAIL*. It doesn’t matter if five or thousand elements are in the list, time for the insertion and deletion will take constant time near the left and right ends.

There are some practical needs to remove elements from the tail or head of a given list. The *BRPOP* and *BLPOP* commands have been introduced from the Redis version 2.0.0 to fulfill the mentioned requirement. In this guide, the *BRPOP* command will be evaluated.

# The BRPOP Command

The *BRPOP* is the advanced version of the *RPOP* command. The *RPOP* command pops an element from the tail of the list stored at a given key. As the name suggests, the BRPOP command is the Blocking version of the RPOP. When the key doesn’t exist, The BRPOP command won’t return with the *nil* value right away as in the RPOP command. Instead, it will wait until a new list has been created at that key. Upon the creation of a new list at the mentioned key, the BRPOP command will pop the tail element. Also, this command accepts multiple keys and only pop elements from the first non-empty key.



## Syntax:

The following is the syntax for the BRPOP command.

| BRPOP list\_key [list\_key ...] timeout |
| --- |

***list\_key***: This is the key of the list.

***timeout***: This is the timeout value in seconds where the client will be blocked until this timeout reaches.

Usually, the BRPOP command returns an array output.

* If a non-empty list has been specified, the command will return the popped element value and key of the containing list as in the following format.

| 1) "CarList" 2) "Benz" |
| --- |

* When no single element is to be popped from any of the lists specified and the timeout has expired, the ***nil*** value will be returned.

| (**nil**) (**10.46s**) |
| --- |

## Example - Radio Station Playlist Manipulation with BRPOP

Let’s assume that a radio station needs to play songs from a daily playlist. The playlist contains song IDs to play. The songs should be ordered in the insertion order. Each song will be picked from the end of the playlist to play.

We can use the Redis list data structure to implement the mentioned playlist and list operations to manipulate the playlist songs. Let’s create a playlist called ***MidNightPlaylist*** and add some song IDs as shown in the following. The *LPUSH* command has been used for that.

| lpush MidNightPlaylist song001 song004 song010 song100 song101 |
| --- |

This would create a list as shown in the following.

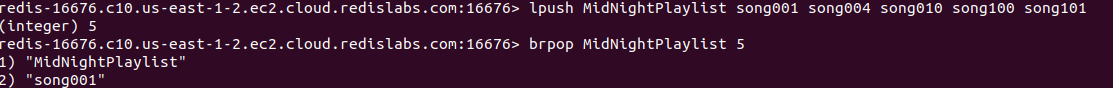
| *head ->* **song101** | **song100** | **song010** | **song004** | **song001** *<- tail* |
| --- |

### Non-blocking Behavior of the BRPOP Command

Now is the time to start a show. We should pick songs from the end of the playlist. Hence, ***song001*** should be removed from the playlist and it has to be played by the recorder. We will be using the BRPOP command to pop *song001* from the playlist and to get the song ID to the client listener.

| **brpop** MidNightPlaylist 5 |
| --- |

The timeout argument has been specified as 5 seconds. Since the *MidNightPlaylist* contains elements, the *BRPOP* command will behave in a non-blocking way. Hence, it will pop and return the tail element to the client.



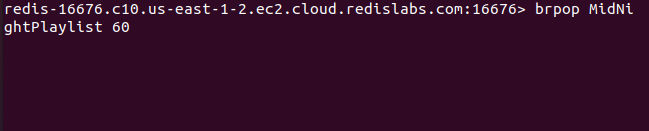
### Blocking Behavior of the BRPOP Command

The above results can be achieved through the RPOP command as well. Hence, let’s look into the real advantage of the BRPOP command with blocking behavior. Let’s remove all the elements from the *MidNightPlaylist* key using the RPOP command.

| rpop MidNightPlaylist 4 |
| --- |

This will remove all the remaining elements from the list and the *MidNightPlaylist* key will be deleted from the Redis key space as well.

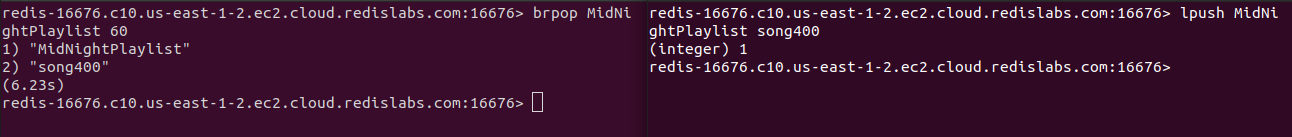
Let’s execute the BRPOP command with 60 seconds timeout and the non-existing key *MidNightPlaylist.* Now the command will behave in a blocking manner. It is waiting for the key to be created and the element to be present in the playlist *MidNightPlaylist.*



Let’s push an element to the MidNightPlaylist via another terminal window that has been connected to the same Redis data store.

| lpush MidNightPlaylist song400 |
| --- |

Upon the creation of the *MidNightPlaylist* list with the element *song400*, the client window which executed the BRPOP command will pop the element *song400* from the playlist and return the song ID as output instantly.



### Pick a Song from Multiple Playlists

There can be multiple playlists created by the radio station. Hence, we should pick a song from the given playlists at a given time. Let’s assume we got three playlists: *MidNightPlaylist1*, *MidNightPlaylist2*, and *MidNightPlaylist3*. Also, the MidNightPlaylist1 is already empty. The other two playlists are non-empty.

| lpush MidNightPlaylist2 song1002 song1005  lpush MidNightPlaylist3 song3000 |
| --- |

Let’s call the BRPOP command with all the three keys as shown in the following.

| brpop MidNightPlaylist1 MidNightPlaylist2 MidNightPlaylist3 10 |
| --- |

Since the first key *MidNightPlaylist1* is empty, it will be ignored by the command here. It will check for the first non-empty key from the available key list. Hence, the command will locate the *MidNightPlaylist2* as the first non-empty key from the order of keys. As expected, the *song1002* is removed from the *MidNightPlaylist2.*



The BRPOP command has constant time complexity near the head and tail when a single key is specified. The time complexity will become O(N) when multiple keys have been specified in the command. Furthermore, this command is very efficient to use in low latency applications such as a queue, stack, timeline in social media, etc.

# Conclusion

To summarize, a Redis list is a collection of string elements stored at a given key while sorted in the insertion order. Several commands are available to operate on Redis lists with constant time complexity near the head and tail. As stated, The BRPOP command is used to remove elements from the right side of the Redis list stored at a given key with the support of blocking. The BRPOP command will block the client connection when no elements are available to remove from the specified lists. As you know, this command accepts multiple keys where an element will pop from the first non-empty list where each key will be checked in the order in which they have been passed to the command.