**Title:** Redis RPOPLPUSH

***Excerpt:***Redis list is a collection of strings where elements can be inserted from both the head and tail. It has constant time complexity in accessing elements near the head and tail which is very efficient to use in low latency applications. Several commands are available to operate on Redis lists. The RPOPLPUSH is one of the widely used commands to implement messaging systems with reliable queues where it removes an element from the tail of the source list and pushes that element to the head of another list.

**Permalink:** redis-rpoplpush

**Category:** Redis

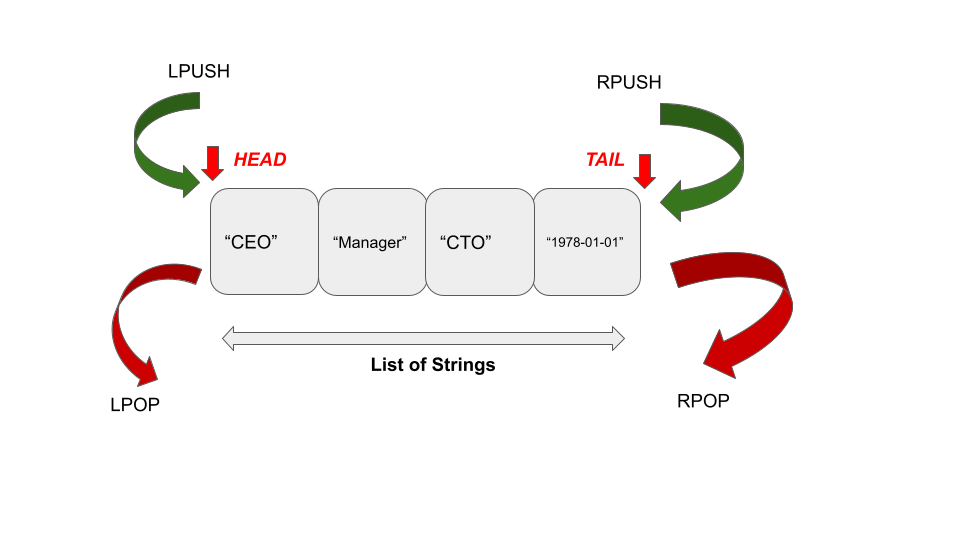
# 

# 

# 

# Introduction to Redis Lists

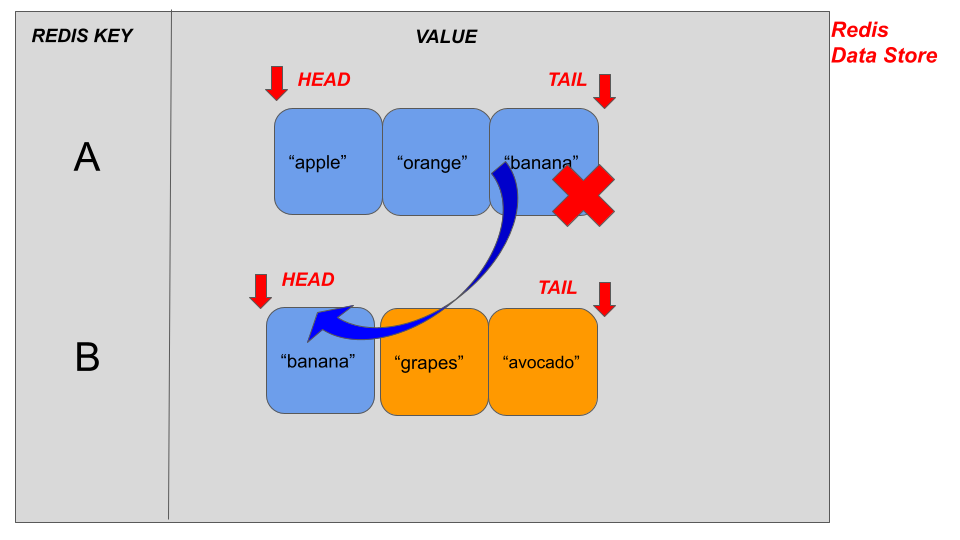
Redis has several in-built data types and each one is for different purposes. The list is one of the widely used data types that can store more than 4 billion strings at a specified Redis key. The special thing about the Redis list is it allows the insertion and removal of elements from both the head and tail which has constant time complexity. The *LPUSH* and *RPUSH* are used to insert string elements near the head and tail of a list. Also, the *LPOP* and *RPOP* remove list elements at the head and tail.



# The RPOPLPUSH Command

The RPOPLPUSH command is used to pop an element from the tail of a source list and insert it at the head of a destination list stored at a given key. Both the removal and insertion at source and destination lists happen simultaneously.

Let’s assume that a list stored at key *A* contains the strings “apple”, “orange”, and “banana”. Furthermore, another list stored at key B has the elements “grapes”, and “avocado”. If we call the RPOPLPUSH command where list A is the source and list B is the destination, it will remove the “banana” string from list A and insert it just before the “grapes” element in list B.



## Syntax:

The RPOPLPUSH command has the following syntax.

| RPOPLPUSH source\_list\_key destination\_list\_key |
| --- |

***Source\_list\_key*** : This is the Redis key of the list that you need to remove an element.

***destination\_list\_key***: This is the Redis key of the list that you need to insert the popped element.

This command returns a string response that is a popped list element from the source list if the source list is available. When the source list doesn’t exist, the command will return *a nil* value. In some cases, the source and destination can be the same. Hence, it will perform both the insertion and removal in the same list.

## Use Case 01 - Implementation of Messaging Queue

Redis lists have been mostly used in messaging server implementations where producers produce data near the head of the list and the consumers consume data from the tail of the list stored at a specified key. Think about a situation where a network failure or a server issue is thrown, the consumer might lose the popped data before the processing begins. It might be a loss that consumers can’t recover in their lifetime. Hence a more reliable queue is needed. Let’s make use of the RPOPLPUSH command to implement a reliability queue.

Let’s create a list *ProcessList* with few elements as shown in the following. The LPUSH command can be used to produce some data into the specified list.

| lpush ProcessList "p1" "p2" "p3" "p4" |
| --- |

The above command will push elements near the head of the list. Hence, the elements will be stored in the following order.

| "P4" | "p3" | "p2" | "p1" |
| --- |

Next, we will be using the *RPOPLPUSH* command as a client to pop the last element of the list ProcessList and push it into a new list stored at the key *ReliabilityQueue.*

| rpoplpush ProcessList ReliabilityQueue |
| --- |

Ideally, the “p1” element has to be popped from the source list and the string “p1” should be the output of this command. At the same time, this element is pushed into the list stored at key *ReliabilityQueue.* Let’s inspect it using the *LRANGE* command.

| lrange ReliabilityQueue 0 5 |
| --- |

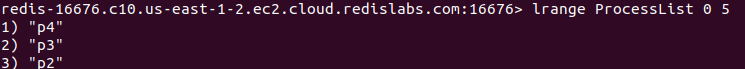
Output:



We will be calling the *LRANGE* command on the list *ProcessList* as well.

| lrange ProcessList 0 5 |
| --- |

Output:



As expected the “p1” element is removed from the source list. The *RPOPLPUSH* command is recommended to use with reliability queues.

## Use Case 02 - Implementation of Reliable Worker Nodes to Process Individual Applications

As mentioned, the source and the destination can be the same for the RPOPLPUSH command. In this case, the insertion and removal are performed in the same list. This can be useful when implementing a monitoring system to process hundreds of interview applications for a given job.

Multiple workers and iterate over the list until all the applications have been processed. The best thing about this kind of circular queue is that it won’t lose any application due to that the item is reinserted into the same list and eventually received by one of the workers in the next iteration.

Let’s create a new list *JobApplicationList*.

| lpush JobApplicationList cv1 cv3 cv7 cv10 |
| --- |

The list should look like the following.

| "cv10" | "cv7" | "cv3" | "cv1" |
| --- |

Now let’s call the RPOPLPUSH command with the same list as the source and destination arguments shown in the following.

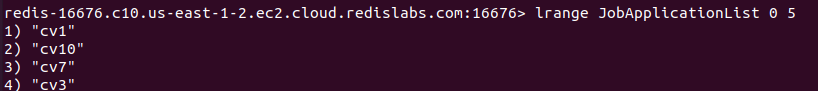
| rpoplpush JobApplicationList JobApplicationList |
| --- |

As expected, the tail element “cv1” has been removed and returned by the command.



We can inspect the *JobApplicationList* list again to check whether the “cv1” element has been inserted near the head of the same list.

| lrange JobApplicationList 0 5 |
| --- |



As you can see, the “cv1” element has been inserted from the head of the list stored at the key *JobApplicationList*. The circular list can be used as a reliable way to implement monitoring systems like this.

# Conclusion

To summarize, the RPOPLPUSH command is one of the widely used commands that has been used to perform both the removal and insertion of lists simultaneously. As you know, this command accepts source and destination lists as arguments. The tail element of the source list will be popped and inserted near the head of the destination list stored at a specified key. As discussed, the RPOPLPUSH command is used to implement reliable queues in messaging systems and also in monitoring systems where multiple worker nodes work on the given list of elements parallel in a reliable way with lost recovery equipped. This command is deprecated from the Redis version 6.2.0 and encouraged to use *the LMOVE* command with the *RIGHT* and *LEFT* arguments.