**Title:** Redis GETEX

***Excerpt:*** The GETEX command behaves almost the same as the GET command in Redis. It supports reading the string values stored at a specified key. At the same time, it is possible to set an expiry time for a key in seconds, milliseconds, and UNIX timestamp format. Additionally, the PERSIST option can be specified to remove a TTL value associated with a given key.

**Permalink:** redis-getex

**Category:** Redis

# 

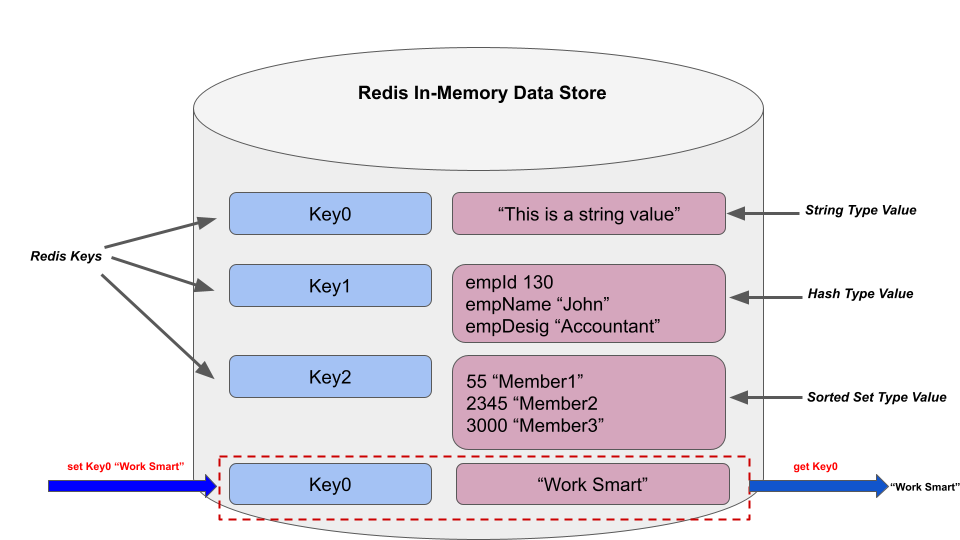
# 

# 

# Redis as a Key-Value Data Store

Redis is a key-value data store that is used as an in-memory cache, database, time series, stream engine, message broker, etc. Since the Redis data store uses a simple key-value pair format to hold data, it is very memory efficient. In addition, the querying speed is very high with low time complexities like O(1) and O(log(N)). Redis keys can hold different data structures such as strings, sets, hashes, sorted sets, and lists.

Usually, the GET and SET commands are used to store and retrieve string-type values stored at a given Redis key. These two commands operate with constant time complexity.



In addition, the GETEX command is available to read string values stored at a given key. But this command is associated with a parallel write operation as well. In this guide, we will be focusing on the GETEX command syntax, its options, and use cases.

# TTL of a Redis Key

The time to live(TTL) value is the expiration time of a given piece of data. Usually, a TTL can be associated with database rows. A time-out can be set with Redis keys as well. So, a Redis client can check the remaining seconds that a given key will be available in the data store. Redis offers a TTL command to check the time to live value of a given key. If there is no timeout associated with a specified key, the TTL command returns -1. We will be using the TTL command in the next sections to check the remaining time of a given key.

# The GETEX Command

As highlighted, the GETEX command is more similar to the GET command except that it can be used to set an expiration time for the specified key. Furthermore, a command option is available to remove a timeout value from a given key. The GETEX command syntax is as follows.

| GETEX key [EX timeout\_in\_seconds | PX timeout\_in\_milliseconds | EXAT timeout\_in\_unix-time-seconds | PXAT timeout\_in\_unix-time-milliseconds | PERSIST] |
| --- |

***key:*** This is the key that points to a string value.

The following optional arguments can be used to set a timeout in seconds or milliseconds.

***EX:*** This will set the expiration time in seconds for a given key.

***PX:*** The key timeout value will be set in milliseconds.

In addition, we can specify a UNIX timestamp value in seconds or milliseconds using the following command options *EXAT* and *PXAT*.

***EXAT:*** The expiration time will be set in the UNIX timestamp format and the timestamp would be in seconds.

***PXAT:*** The timeout for a given Redis key will be set as a UNIX timestamp that is in milliseconds.

***PERSIST:*** By specifying this, you can remove the time to live value associated with a specified key.

The GETEX command returns the value of the specified key. This command only works with string-type values but not with hashes, lists, sorted sets, etc. If you specify a key associated with a non-string value, the command will return an error. In addition, if the key is already expired or doesn’t exist, the command will return ***nil.***

In the following section, we will be covering the practical usage of the GETEX command.

## Use Case - Temporary Reset Link or PIN Code Generation for Password Recovery/ OTPs

A web hosting company is providing a password recovery feature for registered users. As a security measurement, they send a new URL link to the user which will expire within 5 minutes. The company has been using the Redis data store for a long time and they are using the GETEX command to set a time to live value for a given user.

Let’s assume that a user with user\_id 1000 requests a password recovery. So, the recovery URL can be stored at the key *RecoveryURL:UserID:1000*. We can use the SET command to store the recovery URL as follows.

| set RecoveryURL:UserID:1000 "https://abc.com/recovery/userid=1000,key=sdfiuoe3290e9rjs" |
| --- |

Let’s check whether the string URL is properly stored at the key *RecoveryURL:UserID:1000.* We can use the GETEX command as follows.

| getex RecoveryURL:UserID:1000 |
| --- |

The output of the above command is shown in the following.

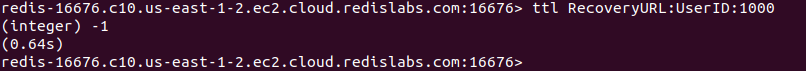


If we have used the GET command instead of the GETEX command, the output will be the same.



Let’s use the TTL command to check whether an expiration time is associated with this key.

| ttl RecoveryURL:UserID:1000 |
| --- |

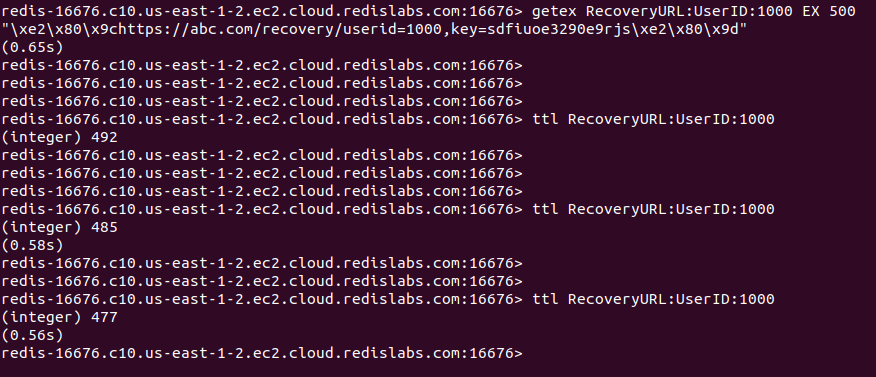


As expected, the -1 has been returned by the TTL command which means no expiry time is associated with the specified key.

As per this example, the hosting company needs to set a 5 minutes timeout for the recovery URL. So, we will be using the *EX* command option. Since 5 minutes is 300 seconds, we can construct the GETEX command as follows.

| getex RecoveryURL:UserID:1000 EX 500 |
| --- |

Let’s run the TTL command a few times to inspect the remaining time that the key will exist in the data store.



As expected, the time is decreasing and eventually the *RecoveryURL:UserID:1000* key will be removed. The following is the output after 5 minutes.



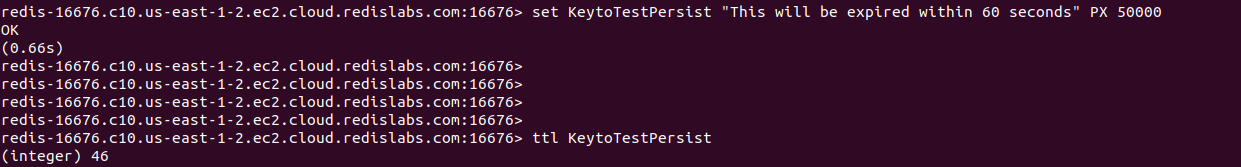
The -2 has been returned by the command which means the key is expired and deleted from the data store.

In some cases, you might need to remove the timeout value associated with a given key. So, the PERSIST command option can be used to delete a TTL value related to a specified key.

Let’s create another sample key and assign a string value to it as follows. In this case, we will be specifying the TTL value in milliseconds by passing the *PX* option.

| set KeytoTestPersist "This will be expired within 60 seconds" PX 50000 |
| --- |

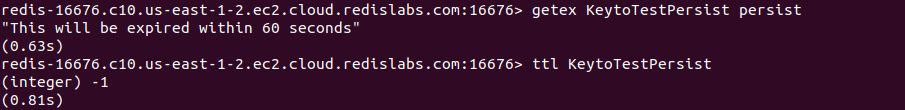
Next, we will be checking the remaining time as follows.



As expected the TTL is decreasing. Let’s call the GETEX command with the PERSIST option as follows.

| getex KeytoTestPersist persist |
| --- |

This would ideally, remove the TTL associated with this key. Let’s inspect the TTL for the key *KeytoTestPersist.*



The TTL command returned the -1 which means no TTL is associated with the key *KeytoTestPersist.*

Similarly, the *GETEX* command can be used to set the expiry time for keys in UNIX timestamp

# Conclusion

In summary, the Redis GETEX command can be used to retrieve the string values stored at a given key as the GET command does. The only difference with the GETEX command is that it supports setting and removing expiry times for a given key. It supports several optional arguments to specify the expiry time in seconds, milliseconds, and UNIX-timestamp seconds or milliseconds. As discussed, the command provides the PERSIST option to remove associated TTL for a given key as well. Overall, the GETEX command operates with constant time complexity.