**Title:** Redis XLEN

**Excerpt:** Redis streams can store multiple entries until the maximum memory is reached. Hence, it provides a way to count the number of entries per stream out of the box. The XLEN command is used to count the number of entries held by a stream stored at a specified key.

**Permalink:** redis-xlen

**Category:** Redis

# 

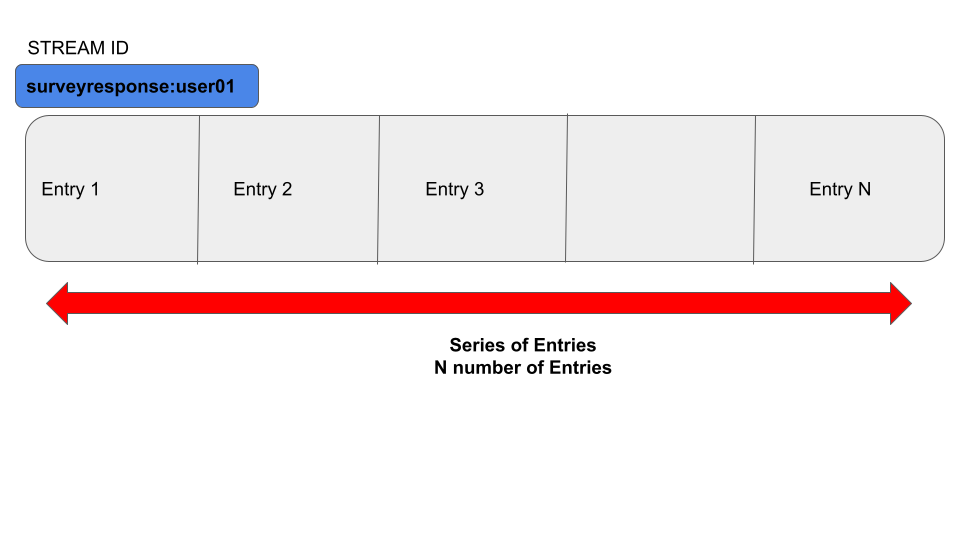
# 

# 

# Redis stream entries

Redis stream is a log data structure that overcomes the limitations of an ordinary append-only log file. This is the newest addition to the Redis data types from version 5.0. It is based on the publisher-consumer concept where multiple sources publish data to the stream and several consumers or consumer groups read from it.

Redis stream can store field-value pairs within its macro nodes. It is very memory efficient and fast in accessing stream entries since it's based on a radix tree structure. The entries look like the Redis hash key-value pairs.



The above illustration shows a stream stored at key *‘surveryresponse:user01’.* It contains N number of entries.

# Count stream entries with XLEN

In real-world applications, we might need to have a clue on how many entries are available in a given stream. It is not practical to make an XRANGE call with minimum and maximum possible IDs to query all the entries per stream and count them programmatically. It consumes time for two operations to read and count programmatically.

Hence, Redis provides an *XLEN* command to count entries for a stream stored at a specified key. This command has constant time complexity which is fast to use in your applications.

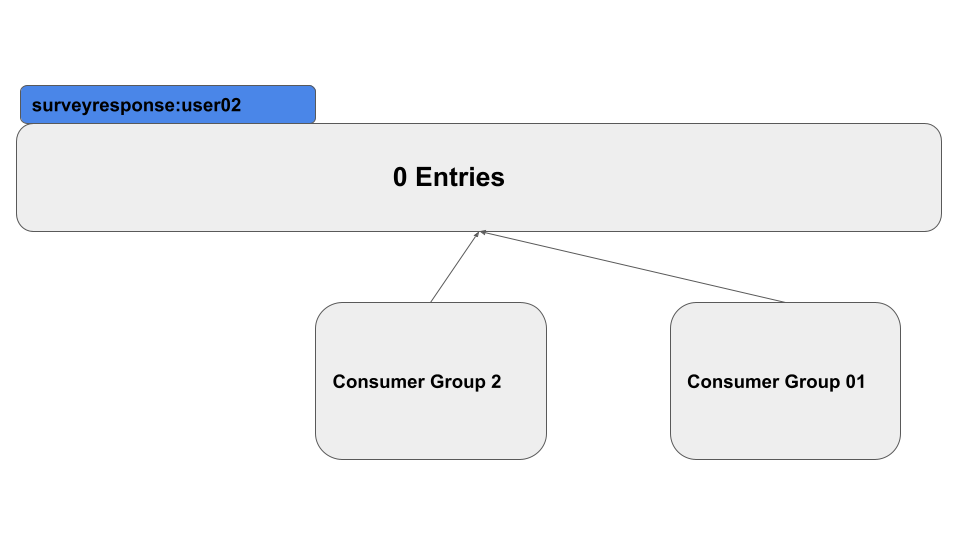
## Syntax

| XLEN <**stream\_key**> |
| --- |

The *XLEN* command returns an integer that is the number of entries stored in a stream.

The streams are a bit different from other Redis data types because a stream can exist with zero entries. Redis streams can have multiple consumer groups attached. The *XDEL* command will not delete the stream itself even if all the entries have been deleted. Hence, the *XLEN* command might return 0 in two cases.

1. Stream is empty or all the entries have been deleted already.
2. Stream key doesn’t exist.



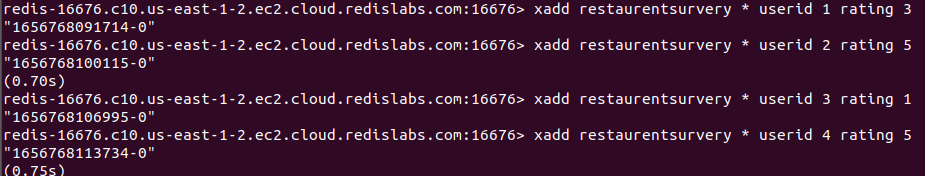
The stream stored at key *surveyresponse:user02* is an empty one. But it has two consumer groups attached. Hence, the stream exists. In real-world applications, You need to do an explicit check with the *EXISTS* command to verify whether the Redis key exists or not.

## Example - Count the number of users participating in an online survey

Let’s assume that a restaurant has conducted an online survey to get a star rating from its customers on their service quality. They have used an in-memory Redis data store to keep track of each customer rating. Each stream entry consists of a couple field-value pairs to store the rating as an integer and user ID as a string.

Let’s create a stream using the *XADD* command.

| xadd restaurentsurvery \* userid 1 rating 3  xadd restaurentsurvery \* userid 2 rating 5  xadd restaurentsurvery \* userid 3 rating 1  xadd restaurentsurvery \* userid 4 rating 5 |
| --- |



Four entries have been added to the stream stored at the key *restaurentsurvey.*

Next, we will be using the *XLEN* command to count the number of users who took part in the survey.

| xlen restaurentsurvery |
| --- |

Output:



As expected, the return value is 4. Assume that there are more than 10000 customers who have participated in the survey. Then the *XLEN* command would be a life saver. Even the Redis XLEN command would take constant time to count the 4 customers or 10000 customers.

# Conclusion

Redis streams can contain thousands of entries until the maximum memory is reached where each entry looks like a field-value pair. Usually, it is cumbersome to count the number of entries stored in a stream programmatically. It consumes time for two operations like reading and counting. Hence, Redis provides count operation out of the box with the XLEN command. It has constant time complexity which is fast to use in high-performance applications. The XLEN command only takes the key of the stream as a parameter.