

### Courses > Redis streams in production

← Previous Next →

## Redis streams in production

## The XAUTOCLAIM command

The **XAUTOCLAIM** command was introduced with Redis 6.2 and further enhanced in Redis 7. It provides a simpler way of claiming messages from a failed consumer and is conceptually equivalent to calling **XPENDING** then **XCLAIM**.

Use **XAUTOCLAIM** to atomically transfer ownership of messages that have been pending for more than a specified number of milliseconds to a new consumer.

Let's see how this works with an example scenario.

First, we'll populate a stream with 200 entries. Each entry will have the same data, but that doesn't matter for this example:

127.0.0.1:6379 > 200 xadd demostream \* hello world

"1679679140391-0"

"1679679140411-0"

"1679679140413-0"

"1679679140434-0"

"1679679140492-0"

"1679679140494-0"

•••

We can verify that 200 entries were created:

127.0.0.1:6379> xlen demostream

(integer) 200



New version available!

Click to Update

Next, let's create a consumer group for this stream, pointing it at the beginning of the stream:

127.0.0.1:6379 × xgroup create demostream democonsumers 0

OK

Our first consumer, **consumer1** now reads some entries from the stream, but does not yet acknowledge them with **XACK**:

127.0.0.1:6379> xreadgroup group democonsumers consumer1 count 3 streams demostream >

- 1) 1) "demostream"
  - 2) 1) 1) "1679679140391-0"
    - 2) 1) "hello"
      - 2) "world"
    - 2) 1) "1679679140411-0"
      - 2) 1) "hello"
        - 2) "world"
    - 3) 1) "1679679140413-0"
      - 2) 1) "hello"
        - 2) "world"

Having read these entries, **consumer1** then crashes, and won't play any further part in this example. This leaves three entries in the pending entries list.

We can now use **XAUTOCLAIM** to claim the first two entries from are still pending and have been idle for at least a minute. We'll

**☆** 

New version available!

127.0.0.1:6379 > xautoclaim demostream democonsumers co

- 1) "1679679140413-0"
  2) 1) 1) "1679679140391-0"
  2) 1) "hello"
  2) "world"
  2) 1) "1679679140411-0"
  2) 1) "hello"
  2) "world"
- 3) (empty array)

When calling **XAUTOCLAIM** here, we pass it the following parameters:

- The name of the stream: demostream
- The name of the consumer group: democonsumers
- The name of the consumer to assign any matching stream entries to: consumer2
- The number of milliseconds that matching stream entries must have been idle for (since the consumer they are currently assigned to last read them): 60000
- The stream entry ID in the pending entries list to start from, here we provide 0 to start at the beginning of the PEL: 0
- An optional COUNT clause specifying the maximum number of entries to re-assign if enough are found that meet the criteria. Here we're specifying 2. The default if nothing is specified is 100

**XAUTOCLAIM** responds with an array type response containing three elements:

- 1. A stream entry ID to use as the start point for subsequent calls to **XAUTOCLAIM**. This is **1679679140413-0** in the above example. We can use this to iterate through the pending entries list.
- 2. An array containing all of the entries claimed, including their IDs and payloads.
- 3. (Redis 7 only): An array containing any message IDs that were in the pending entries list but had been deleted from the stream, and which have now been cleaned up from the pending entries list.

Note that calling **XAUTOCLAIM** in this way also increments the each matching entry returned.



New version available!

Using XINFO CONSUMERS, we can now see that two entries have to consumer2:	re been re-assigned	
127.0.0.1:6379> xinfo consumers demostream democonsumer	rs	
1) 1) "name"		
2) "consumer1"		
3) "pending"		
4) (integer) 1		
5) "idle"		
6) (integer) 1268434		
2) 1) "name"		
2) "consumer2"		
3) "pending"		
4) (integer) 2		
5) "idle"		
6) (integer) 901664		
Finally, let's look at another optional modifier to the <b>XAUTOCLAIM</b> command.		
Here we'll use the entry ID returned from the previous call to XX tries and assign them to consumer3:	AUTOCLAIM to claim more en-	
127.0.0.1:6379> xautoclaim demostream democonsumers con 1679679140413-0 count 10 justid	nsumer3 60000	
1) "0-0"		
2) 1) "1679679140413-0"		
3) (empty array)	New version available!	

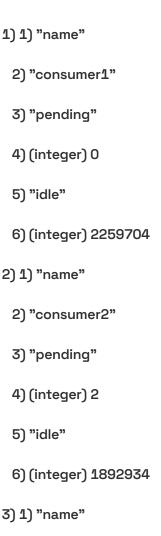
The JUSTID modifier changes the behavior of XAUTOCLAIM in two ways:

- 1. Only the matching entry IDs are returned, the payloads are not. This can be useful to save network bandwidth if the calling application does not need the payload.
- 2. The attempted delivery count for each matching entry is not updated.

In the example above, we have also exhausted matching entries in the pending entries list at this time, so the next ID returned is **0-0**.

Finally, we can re-run the XINFO CONSUMERS command to verify that consumer3 now has entries assigned to it, and that consumer1 has had all entries that were assigned to it moved elsewhere:

#### 127.0.0.1:6379> xinfo consumers demostream democonsumers



2) "consumer3"

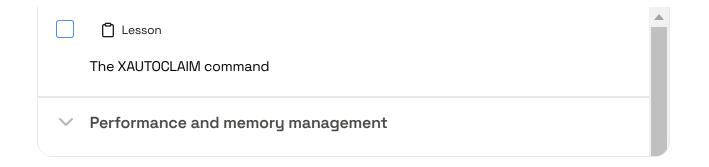
3) "pending"

4) (integer) 1



New version available!

5) '	'idle"		
6) (	integer) 125686		
For n	nore information, refer to the <b>XAUTOCLAIM</b> command pag	e on redis.io.	
← Pre	vious	Next -	>
Mo	dules	<b>&gt;&gt;</b>	
~	Course overview		
	Lesson		
	Course overview		
	🖰 Lesson		
	Environment setup		
~	Advanced consumer group management		
	C Lesson		
	Managing pending messages		
	✓ Assessment		
	Quiz 1   Redis streams in production		
	Lesson		
	Consumer recovery & poison-pill messages		
	✓ Assessment	New version available!	
	Quiz 2   Redis streams in production	<b>₹</b>	



# Redis

Cloud

Software

Pricing

Support

University Feedback

University Help

Contact us

Legal notices

f

in

(O)

 $\mathbb{X}$ 

Trust Terms of use Privacy policy