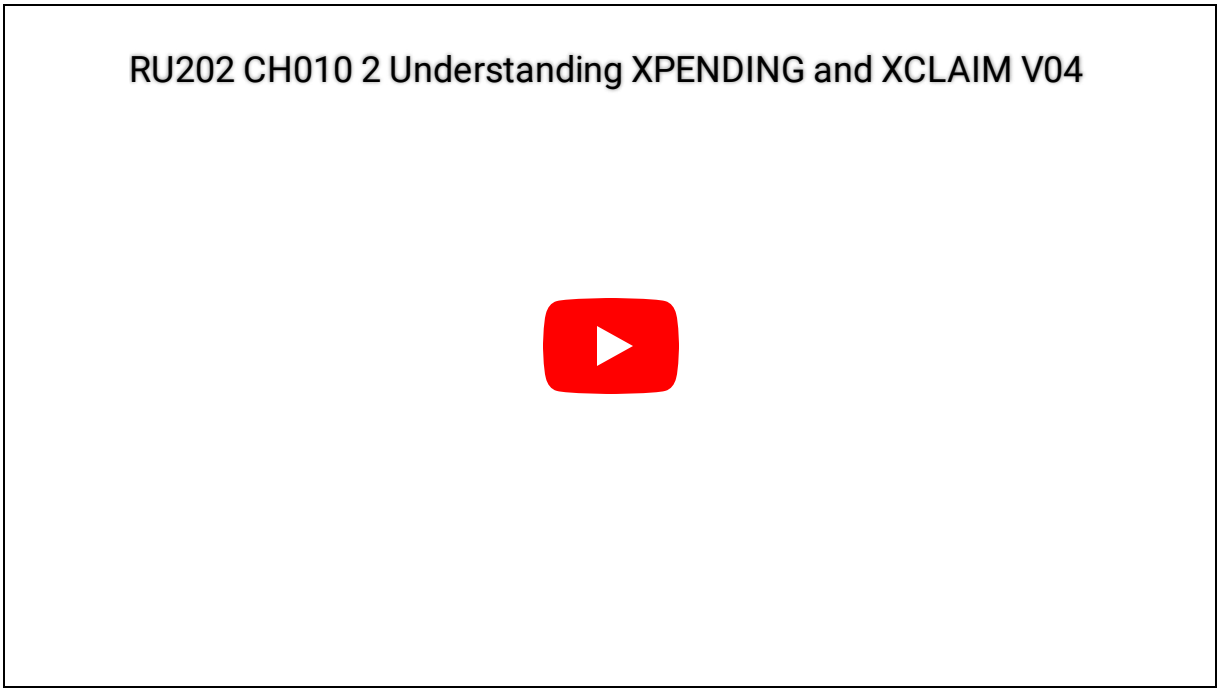
Managing pending messages

Documentation at redis.io

XPENDING command
XCLAIM command
XINFO command



XPENDING Enhancements in Redis 6.2

Redis 6.2 introduced two useful enhancements to the **XPENDING** command. Let's take a moment to look at these.

First, check your Redis server version with the **INFO SERVER** command:

```
127.0.0.1:6379> INFO SERVER
```

```
# Server

redis_version:7.0.8

redis_git_sha1:00000000

redis_git_dirty:0

redis_build_id:c869ebfd8f51f71c

redis_mode:standalone

...
```

Here, I'm running Redis 7.0.8 so we're good to go!

Idle Time Filter

From Redis 6.2 onwards, the **IDLE** modifier can be used with the **XPENDING** command. This allows us to filter entries in the Pending Entry List that have been pending for at least a specified number of milliseconds. For example, let's create a stream with a consumer group and read some entries without acknowledging them:

```
127.0.0.1:6379> 10 XADD mystream * hello world

"1677608606044-0"

"1677608606046-0"

"1677608606048-0"

"1677608606049-0"

"1677608606050-0"

"1677608606052-0"

"1677608606053-0"

"1677608606054-0"

"1677608606056-0"

"1677608606057-0"
```

```
127.0.0.1:6379> XGROUP CREATE mystream consumers 0
OK
127.0.0.1:6379> XREADGROUP GROUP consumers consumer1 COUNT 2 STREAMS mystream
>
1) 1) "mystream"
   2) 1) 1) "1677608606044-0"
         2) 1) "hello"
            2) "world"
      2) 1) "1677608606046-0"
         2) 1) "hello"
            2) "world"
127.0.0.1:6379> XREADGROUP GROUP consumers consumer2 COUNT 1 STREAMS mystream
>
1) 1) "mystream"
   2) 1) 1) "1677608606048-0"
         2) 1) "hello"
            2) "world"
```

After a few seconds, let's read another entry - again without acknowledging it:

```
127.0.0.1:6379> XREADGROUP GROUP consumers consumer2 COUNT 1 STREAMS mystream
>
1) 1) "mystream"
   2) 1) 1) "1677608606049-0"
         2) 1) "hello"
            2) "world"
```

Once a few more seconds have passed, let's use **XPENDING** to see which entries were read more than 15 seconds ago and have not yet been acknowledged:

```
127.0.0.1:6379> XPENDING mystream consumers IDLE 15000 - + 10
1) 1) "1677608606044-0"
   2) "consumer1"
   3) (integer) 125153
   4) (integer) 1
2) 1) "1677608606046-0"
   2) "consumer1"
   3) (integer) 125153
   4) (integer) 1
3) 1) "1677608606048-0"
   2) "consumer2"
   3) (integer) 112402
   4) (integer) 1
```

Note that the entry with ID **1677608606049-0** is not present in the output, at it hasn't been idle for 15 seconds yet.

We can also specify the name of a consumer in the group to see only output for that specific consumer:

```
127.0.0.1:6379> XPENDING mystream consumers IDLE 15000 - + 10 consumer1
1) 1) "1677608606044-0"
   2) "consumer1"
   3) (integer) 203159
```

4) (integer) 1

2) 1) "1677608606046-0"

2) "consumer1"

3) (integer) 203159

4) (integer) 1

Exclusive Ranges

Redis 6.2 also added the ability to specify an exclusive range of IDs to **XPENDING**. For example, if we have iterated the pending entries list as far as the entry whose ID timestamp portion is **1677608606046**, then we could ask for the rest of the list like this:

127.0.0.1:6379> XPENDING mystream consumers 1677608606046 + 10

1) 1) "1677608606046-0"

2) "consumer1"

3) (integer) 1021150

4) (integer) 1

2) 1) "1677608606048-0"

2) "consumer2"

3) (integer) 1008399

4) (integer) 1

3) 1) "1677608606049-0"

2) "consumer2"

3) (integer) 937004

4) (integer) 1

Note that the entry ID that we provided in the **XPENDING** command is returned in the response from Redis. If we want only entries whose IDs are higher than the one provided, Redis 6.2 or higher allows us to do this by preceding the entry ID with (:

```
127.0.0.1:6379> XPENDING mystream consumers (1677608606046 + 10
1) 1) "1677608606048-0"
   2) "consumer2"
   3) (integer) 1020721
   4) (integer) 1
2) 1) "1677608606049-0"
   2) "consumer2"
   3) (integer) 949326
   4) (integer) 1
```

In this case, we now only receive entries whose IDs are higher than the one provided.

Further Resources

For more information, see the **XPENDING** command page on redis.io.

← Previous

Next →

Modules

Quiz 2 | Redis streams in production

Lesson

The XAUTOCLAIM command

Performance and memory management

Lesson

Performance considerations

Assessment

Quiz 3 | Redis streams in production

Lesson

Stream memory usage

Assessment

Quiz 4 | Redis streams in production

Lesson

Stream capping strategies

Legal notices





Trust Terms of use Privacy policy