Sheet - Implementing Expiration (TTL) of Keys in Redis

Redis provides built-in support for key expiration using TTL (Time To Live). This feature is widely used for caching, session management, OTPs, and temporary data storage.

1. Setting TTL while creating a key

You can define expiration at the time of key creation using SET with EX or PX.

Example:

SET session:user123 "logged_in" EX 60

EX → seconds

PX → milliseconds

```
127.0.0.1:6379> SET session:user123 "logged_in" EX 60
OK
127.0.0.1:6379>
```

2. Setting TTL on an existing key

Use the EXPIRE command.

Example:

SET cache:data "temporary"

EXPIRE cache:data 120

```
127.0.0.1:6379> SET cache:data "temporary"
OK
127.0.0.1:6379> EXPIRE cache:data 120
(integer) 1
127.0.0.1:6379>
```

3. Checking remaining TTL

Use TTL or PTTL.

TTL cache:data

PTTL cache:data

```
127.0.0.1:6379> TTL cache:data
(integer) 62
127.0.0.1:6379>
```

Return values:

> 0 → Remaining time

-1 → Key exists but no expiration

-2 → Key does not exist


```
127.0.0.1:6379> TTL cache:data
(integer) 62
127.0.0.1:6379>
```


4. Removing expiration

Use PERSIST to remove TTL.

Example:

PERSIST cache:data

```
127.0.0.1:6379> PERSIST cache:data
(integer) 0
```

5. Updating TTL

Calling EXPIRE again resets the TTL.

Example:

EXPIRE cache:data 300

```
127.0.0.1:6379> EXPIRE cache:data 300
(integer) 1
127.0.0.1:6379> TTL cache:data
(integer) 298
```

6. Expiring keys at a specific time

Use EXPIREAT or PEXPIREAT with Unix timestamps.

Example:

EXPIREAT report:data 1735600000

```
127.0.0.1:6379> EXPIREAT report:data 1735600000
(integer) 0
```

7. Real-world example: Session expiration

SET user:session:101 "active" EX 1800

```
127.0.0.1:6379> SET user:session:101 "active" EX 1800
OK
```

8. Behavior after expiration

Expired keys are automatically deleted by Redis. Accessing them returns nil.

Summary

- SET key value EX seconds → Set key with TTL
- EXPIRE key seconds → Add or update TTL
- TTL key → Check TTL
- PERSIST key → Remove TTL
- EXPIREAT key timestamp → Expire at fixed time