100 XP

Exercise - Broker messages using Streams

15 minutes

This module requires a sandbox to complete. A **sandbox** gives you access to free resources. Your personal subscription will not be charged. The sandbox may only be used to complete training on Microsoft Learn. Use for any other reason is prohibited, and may result in permanent loss of access to the sandbox.

Microsoft provides this lab experience and related content for educational purposes. All presented information is owned by Microsoft and intended solely for learning about the covered products and services in this Microsoft Learn module.

Sign in to activate sandbox

Your retail application currently uses server-side files to record pertinent information from mobile devices when running in-house software. Unfortunately, there have been multiple issues with scaling this solution out; including concurrency between numerous clients, missing data, and locks preventing reading or writing actions. Your team has decided to move forward with a middleware solution that can serve as a message broker.

In this exercise, implement Streams in an Azure Cache for Redis instance.

Add entries to a stream

Entries are added to a new or existing stream using the `XADD` command. The stream is automatically created if it doesn't already exists.

1. Sign in to the Azure portal using the same account you used to activate the sandbox.
2. Within the **Azure services** category, select **More services**, select the **Databases** category and then select **Azure Cache for Redis**.
3. Select the Azure Cache for Redis instance you created in a previous exercise.
4. In the resource pane, select **Console** to open the Redis console.
5. In the console, use the `XADD` command to add the following new entries:

Expand table

Stream	Key	Data
org.logs.clientapp	1324092248593-0	`device-id mobile error unknown-crash`
org.logs.clientapp	1481945061467-0	`worker-process 1788 status success`

Redis

```
XADD org.logs.clientapp 1324092248593-0 device-id mobile error un-
known-crash

XADD org.logs.clientapp 1481945061467-0 worker-process 1788 status
success
```

6. Observe the output from the two invocations of the `XADD` command. The output includes the key of the newly added entries.

Redis

```
"1324092248593-0"

"1481945061467-0"
```

7. Use the `XADD` command to add another new entry with an automatically generated identifier:

Expand table

Stream	Data
org.logs.clientapp	`application-status started`

Redis

```
XADD org.logs.clientapp * application-status started
```

8. Observe the output from the invocation of the `XADD` command. The output includes a newly generated key based on the current Unix time in milliseconds and a sequence

number. For example, if the key is **1638502526759-0**, then the output would be:

```
Redis

"1638502526759-0"
```

Retrieve and count all entires in a stream

The `XLEN` command counts the number of entries in a stream. Once you're ready to query the entries, you can use the `XRANGE` command to get entries within the stream.

1. Use the `XLEN` command to count the number of entries in the **org.logs.clientapp** stream:

   ```
   Redis

   XLEN org.logs.clientapp
   ```

2. Observe the output from the `XLEN` command. The output is an integer with a value of **3** for the entries created earlier in this exercise.

   ```
   Redis

   (integer) 3
   ```

3. Use the `XRANGE` command and both the `+` and `-` operators to get a range of all data in the **org.logs.clientapp** stream:

   ```
   Redis

   XRANGE org.logs.clientapp - +
   ```

4. Observe the output from invoking the `XRANGE` command. This output includes all three entries in the stream.

   ```
   Redis

   1) 1) "1324092248593-0"
      2) 1) "device-id"
         2) "mobile"
         3) "error"
         4) "unknown-crash"
   2) 1) "1481945061467-0"
      2) 1) "worker-process"
         2) "1788"
   ```

```
      3) "status"
      4) "success"
3) 1) "1638502526759-0"
   2) 1) "application-status"
      2) "started"
```

> ⓘ Note
>
> Your last key will not exactly match the identifier used in this example.

Retrieve a subset of entries in a stream

The `XRANGE` command includes a `+` and `-` operator. These operators can be used with keys to query a subset of the data in a stream based on a time range.

1. Invoke the `XRANGE` command using the `-` operator and the key of the second entry (**1481945061467-0**):

   Redis

   ```
   XRANGE org.logs.clientapp - 1481945061467-0
   ```

2. Observe the output of the invocation of the `XRANGE` command. The output includes all entries from the start of the stream, chronologically, up to the second entry.

   Redis

   ```
   1) 1) "1324092248593-0"
      2) 1) "device-id"
         2) "mobile"
         3) "error"
         4) "unknown-crash"
   2) 1) "1481945061467-0"
      2) 1) "worker-process"
         2) "1788"
         3) "status"
         4) "success"
   ```

3. Invoke the `XRANGE` command using the key of the second entry (**1481945061467-0**) and the `+` operator:

   Redis

```
XRANGE org.logs.clientapp 1481945061467-0 +
```

4. Observe the output of the invocation of the `XRANGE` command. The output includes the second entry, and then all entries up to the end of the stream, chronologically.

Redis

```
1) 1) "1481945061467-0"
   2) 1) "worker-process"
      2) "1788"
      3) "status"
      4) "success"
2) 1) "1638502526759-0"
   2) 1) "application-status"
      2) "started"
```

> **Note**
>
> The last key will not exactly match the one used in this example.

Next unit: Knowledge check

Continue >