Streams REST API

Module Objectives

In this module, you will look at the Continuuity Reactor Streams REST API:

- Creating a Stream
- Sending Events to a Stream
- Reading Events from a Stream
- Reading Multiple Events from a Stream

Streams Recap

Stream REST API supports:

- Creating Streams
- Sending events to a Stream
- Reading single events from a Stream
- Streams may have multiple consumers (for example, multiple Flows), each of which may be a group of different agents (for example, multiple instances of a Flowlet)
- In order to read events from a Stream, a client application must first obtain a consumer (group) id, which is then passed to subsequent read requests

---

Creating a Stream

A Stream can be created with an HTTP PUT method to the URL:

```
PUT <base-url>/streams/<new-stream-id>
```

Parameter

`<new-stream-id>`

Description

Name of the Stream to be created

HTTP Responses

Status Code : ***Description***

`200 OK` : The event either successfully created a Stream or the Stream already exists

---

Creating a Stream: Example

`PUT <base-url>/streams/mystream`

- The `<new-stream-id>` (*mystream*) should only contain ASCII letters, digits and hyphens
- If the Stream already exists, no error is returned, and the existing Stream remains in place

---

Sending Events to a Stream

An event can be sent to a Stream by an HTTP POST method to the URL of the Stream:

```
POST <base-url>/streams/<stream-id>
```

Parameter

`<stream-id>`

Description

Name of an existing Stream

HTTP Responses

Status Code : ***Description***

`200 OK` : The event was successfully received `404 Not Found` : The Stream does not exist

Note: The response will always have an empty body

Sending Events to a Stream: Example

```
POST <base-url>/streams/mystream
```

- The body of the request must contain the event in binary form
- Pass headers for the event as HTTP headers, prefixing them with the *stream-id*: `<stream-id>.<property>:<string value>`

After receiving the request, the HTTP handler transforms it into a Stream event:

1. The body of the event is an identical copy of the bytes found in the body of the HTTP post request
2. If the request contains any headers prefixed with the *stream-id*, the *stream-id* prefix is stripped from the header name and the header is added to the event

---

Reading Events from a Stream: Getting a Consumer-ID

Get a *Consumer-ID* for a Stream by sending an HTTP POST method to the URL:

```
POST <base-url>/streams/<stream-id>/consumer-id
```

Parameter

`<stream-id>`

Description

Name of an existing Stream

HTTP Responses

Status Code : *Description*

`200 OK` : The event was successfully received and a new `consumer-id` was returned `404 Not Found` : The Stream does not exist

---

Reading Events from a Stream: Getting a Consumer-ID: Example

`POST <base-url>/streams/mystream/consumer-id`

Requests a `Consumer-ID` for the Stream named *mystream*

Reading Events from a Stream

- Streams may have multiple consumers (for example, multiple Flows), each of which may be a group of different agents (for example, multiple instances of a Flowlet)
- In order to read events from a Stream, a client application must first obtain a consumer (group) id, which is then passed to subsequent read requests
- The `Consumer-ID` is returned in a response header and in the body of the response: `X-Continuuity-ConsumerId: <consumer-id>`
- Once you have the `Consumer-ID`, single events can be read from the Stream

---

Reading Events from a Stream: Using the Consumer-ID

A read is performed as an HTTP POST method to the URL:

```
POST <base-url>/streams/<stream-id>/dequeue
```

Parameter

`<stream-id>`

Description

Name of the Stream to be read from

The request must pass the `Consumer-ID` in a header of the form:

```
X-Continuuity-ConsumerId: <consumer-id>
```

HTTP Responses

Status Code : *Description*

`200 OK` : The event was successfully received and the result of the read was returned `204 No Content` : The Stream exists but it is either empty or the given `Consumer-ID` has read all the events in the Stream `404 Not Found` : The Stream does not exist

---

Reading Events from a Stream: Using the Consumer-ID: Example

`POST <base-url>/streams/mystream/dequeue`

Read the next event from an existing Stream named *mystream*

Comments

- The read will always return the next event from the Stream that was inserted first and has not been read yet (first-in, first-out or FIFO semantics)
- If the Stream has never been read from before, the first event will be read
- You can always start reading from the first event by getting a new `Consumer-ID`

Reading Events from a Stream

For example, in order to read the third event that was sent to a Stream, two previous reads have to be performed after receiving the `Consumer-ID`

The response will contain the binary body of the event in its body and a header for each header of the Stream event, analogous to how you send headers when posting an event to the Stream:

```
<stream-id>.<property>:<value>
```

Reading Multiple Events from a Stream

Reading multiple events is not supported directly by the Stream HTTP API, but the command-line tool `stream-client` demonstrates how to view *all*, the *first N*, or the *last N* events in the Stream.

For more information, see the Stream Command Line Client `stream-client` in the `/bin` directory of the Continuuity Reactor SDK distribution.

Run at the command line

```
$ stream-client --help
```

for usage and documentation of options.

Module Summary

You should now be able to:

- Create a Stream
- Send Events to a Stream
- Read Events from a Stream
- Read Multiple Events from a Stream

Module Completed

Chapter Index