GSN Data Distributer

In GSN, a Data Distributer enables a client to receive the data from a virtual sensor in a PUSH fashion. The client registers a query to the Data Distributer which will send both historical and live data back to the client. GSN internally embeds the following 3 different instances of Data Distributer. This document describes both the *Connection oriented* and *Connectionless* Data Distributer as they can by used by a remote client to get data out of GSN.

1. **Local Data Distributer**: This Data Distributer can be used only by clients running in the same process as GSN and thus is the cornerstone for the implementation of the local wrapper.
2. **Connection Oriented Data Distributer**: This Data Distributer can be accessed remotely through a HTTP calls. It is the cornerstone for the implementation of the Rest Remote Wrapper.
3. **Connectionless Data Distributer**: This Data Distributer can be accessed remotely through HTTP calls. It is the cornerstone for the implementation of the PUSH Remote Wrapper.

The following Sequence Diagram describes the high level process between GSN and a client to send data. When registering a query, a client defines the **startTime** parameter for which the data are downloaded. Once all the historical data have been sent, GSN will keep sending the live data as they arrive from the sources.

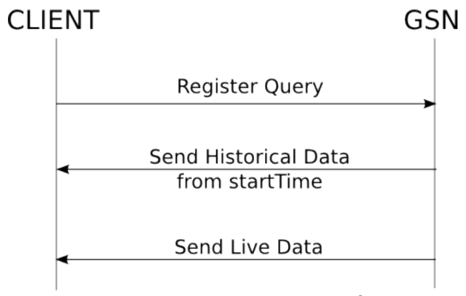

Illustration 1: Sequence diagram for Data Distributer data transfert

Connection Oriented Data Distributer

This Data Distributer keeps the TCP connection (originated when the client register its query through HTTP) alive, in order to send the data back to the client. This Data Distributer has the advantage of limiting the number of connections establishment and thus should be used for high rate data sources. Moreover, this Data Source is useful for clients behind a firewall or NAT as no connection is needed from GSN to the client.

Concerning the implementation, this Data Distributer is based on the Continuation feature of Jetty Web Server which provides asynchronous servlets[1].

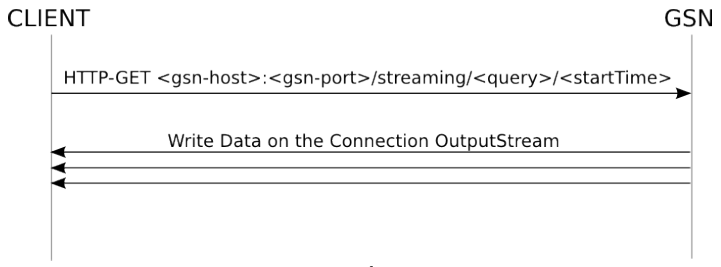
CLIENT
GSN
HTTP-GET <gsn-host>:<gsn-port>/streaming/<query>/<startTime>
Write Data on the Connection OutputStream

Illustration 2: Sequence diagram for the connection oriented Data Distributer

- **gsn-host** The host address of the machine running the GSN instance.
- **gsn-port** The GSN port.
- **query** This is a sql query sent to perform filtering at the source. This query can only contain one single table (no joins, self-joins, inner-selects) and basic filtering (no aggregation, group by, having, database specific key words.
- **startTime** The system starts downloading the historical data. Once the historical items delivered, the data producer sends the real time data as they get produced.

Example

In this example, we are getting data from the *MemoryMonitorVS* virtual sensor running in a GSN instance on 128.178.151.93:22001. Thus the query becomes:

- **gsn-host** 128.178.151.93
- **gsn-port** 22001
- **query** select * from MemoryMonitorVS
- **startTime** 2009-02-06T11:56:10.004+02:00

The HTTP URL to execute the GET query becomes:

```
http://128.178.151.93:22001/streaming/select%20*%20from%20MemoryMonitorVS/2009-02-06T11%3A56%3A10.004%2B02%3A00
```

Note that the URL has to be *percent encoded*[2]. As a test, try an online tool such as: http://meyerweb.com/eric/tools/dencoder/. This URL can be then used from any HTTP client such as a standard web browser or the curl command.

```
curl http://128.178.151.93:22001/streaming/select%20*%20from%20MemoryMonitorVS/2009-02-06T11%3A56%3A10.004%2B02%3A00
```

Executing this query should return something like the following:

1 http://wiki.eclipse.org/Jetty/Feature/Continuations
2 http://en.wikipedia.org/wiki/Percent-encoding

```
<object-stream>
  <strcture-array>
    <strcture>
      <description>Not Provided</description>
      <name>HEAP</name>
      <dataTypeID>5</dataTypeID>
      <type>double</type>
    </strcture>
    <strcture>
      <description>Not Provided</description>
      <name>NON_HEAP</name>
      <dataTypeID>5</dataTypeID>
      <type>double</type>
    </strcture>
    <strcture>
      <description>Not Provided</description>
      <name>PENDING_FINALIZATION_COUNT</name>
      <dataTypeID>5</dataTypeID>
      <type>double</type>
    </strcture>
  </strcture-array>
  <stream-element timestamp="2010-05-03 14:28:24.947 CEST">
    <field name="HEAP" type="numeric">7110392.0</field>
    <field name="NON_HEAP" type="numeric">3.0026592E7</field>
    <field name="PENDING_FINALIZATION_COUNT" type="numeric">0.0</field>
  </stream-element>
  <stream-element timestamp="2010-05-03 14:28:35.545 CEST">
    <field name="HEAP" type="numeric">7255048.0</field>
    <field name="NON_HEAP" type="numeric">3.064324E7</field>
    <field name="PENDING_FINALIZATION_COUNT" type="numeric">0.0</field>
  </stream-element>
  <stream-element timestamp="2010-05-03 14:28:45.783 CEST">
    <field name="HEAP" type="numeric">7342776.0</field>
    <field name="NON_HEAP" type="numeric">3.0676648E7</field>
    <field name="PENDING_FINALIZATION_COUNT" type="numeric">0.0</field>
  </stream-element>
  <stream-element timestamp="2010-05-03 14:28:55.848 CEST">
    <field name="HEAP" type="numeric">7522352.0</field>
    <field name="NON_HEAP" type="numeric">3.0699752E7</field>
    <field name="PENDING_FINALIZATION_COUNT" type="numeric">0.0</field>
  </stream-element>

  . . .
```

Connectionless Data Distributer

This Data Distributer makes a new HTTP connection for each data to be sent, from GSN to the client. Thus with this Data Distributer, the client need to embed a HTTP server. This implementation is useful for low data rate sources as they limit the resources needed to maintain a connection alive. Note that the data sent to the client has to be acknowledged through the HTTP status. A 200 HTTP status must be returned upon a successful data reception whereas a 300 HTTP status must be returned for any transmission failure.

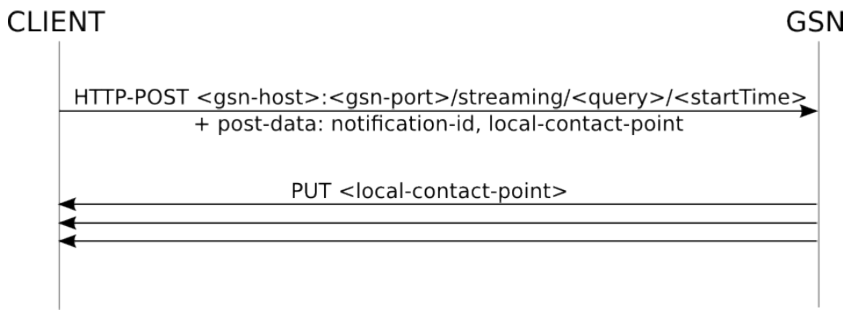
CLIENT
GSN
HTTP-POST <gsn-host>:<gsn-port>/streaming/<query>/<startTime>
+ post-data: notification-id, local-contact-point
PUT <local-contact-point>

Illustration 3: Sequence diagram for the connectionless Data Distributer

- **gsn-host** The host address of the machine running the GSN instance.
- **gsn-port** The GSN port.
- **query** This is a sql query sent to perform filtering at the source. This query can only contain one single table (no joins, self-joins, inner-selects) and basic filtering (no aggregation, group by, having, database specific key words.
- **startTime** The system starts downloading the historical data. Once the historical items delivered, the data producer sends the real time data as they get produced.
- **local-contact-point** The client host and port that the Data Distributer will use to send the data.
- **notification-id** a random number used to identify the client registering the query.

Example

In this example, we are getting data from the *MemoryMonitorVS* virtual sensor running in a GSN instance on 128.178.151.93:22001. In order to get the data, we have a TCP server running locally on localhost:22222. Thus the query becomes:

- **gsn-host** 128.178.151.93
- **gsn-port** 22001
- **query** select * from MemoryMonitorVS
- **startTime** 2009-02-06T11:56:10.004+02:00
- **local-contact-point** localhost:22222
- **notification-id** 123456 (randomly chosen)

As an example, we are running the netcat command in server mode so that it can receive the data from GSN. Morever, we are using the wget command to issue the query registration. Note that this example will return only one element.

```
# 1. Run the netcat server in server mode with the command:
nc -l -p 22222 -v # listening at local port 22222

# 2. Then we register the query with the following command:
wget --post-data "notification-id=123456&local-contact-point=http://localhost:22222"
"http://128.178.151.93:22001/streaming/select * from MemoryMonitorVS/" -O -
```

Then you should see the data output structure in the terminal in which you ran the wget command. If you check the netcat's terminal, you should see one stream element (formatted in URL encoded form) delivered such as the one below:

```
notification-id=123456.0&data=%3Cstream-element+timestamp%3D%222010-05-03+17%3A13%3A07.52+CEST%22%3E%0A++%3Cfield+name%3D%22HEAP
%22+type%3D%22numeric%22%3E1.0116944E7%3C%2Ffield%3E%0A++%3Cfield+name%3D%22NON_HEAP%22+type%3D%22numeric%22%3E3.3160752E7%3C
%2Ffield%3E%0A++%3Cfield+name%3D%22PENDING_FINALIZATION_COUNT%22+type%3D%22numeric%22%3E0.0%3C%2Ffield%3E%0A%3C%2Fstream-element
%3E
```