Google Analytics API

Analytics Data API Overview

The Google Analytics Data API v1 gives you programmatic access to Google Analytics 4 (GA4) report data.

Note, this API will not work with the Universal Analytics (UA)

Sample program to request data from the Google Analytics Data API v1

Follow the instructions here to create a sample java program to pull data for your site. Note, the program below was in Eclipse.

- https://developers.google.com/analytics/devguides/reporting/data/v1/quickstart-client-libraries
- Additionally, the sample program will need to access to the environment variable (`GOOGLE_APPLICATION_CREDENTIALS`)`set from above. In Eclipse, under the run configuration, go to Environment tab and enter the variable and directory.`

QuickstartSample

```
package com.example.analytics;

import java.util.Iterator;
import java.util.List;

import com.google.analytics.data.v1beta.BetaAnalyticsDataClient;
import com.google.analytics.data.v1beta.DateRange;
import com.google.analytics.data.v1beta.Dimension;
import com.google.analytics.data.v1beta.DimensionHeader;
import com.google.analytics.data.v1beta.Metric;
import com.google.analytics.data.v1beta.MetricHeader;
import com.google.analytics.data.v1beta.Row;
import com.google.analytics.data.v1beta.RunReportRequest;
import com.google.analytics.data.v1beta.RunReportResponse;

/*

To run this sample using Maven:
  cd java-analytics-data/samples/snippets
  mvn compile
  mvn exec:java -Dexec.mainClass="com.example.analytics.QuickstartSample"

 */

public class QuickstartSample {

          public static void main(String... args) throws Exception {
            /**
             * TODO(developer): Replace this variable with your Google Analytics 4 property ID before
             * running the sample.
             */

//            String propertyId = "269896522";  // dev
            String propertyId = "263363085";  // prod
            sampleRunReport(propertyId);
          }

          // This is an example snippet that calls the Google Analytics Data API and runs a simple report
          // on the provided GA4 property id.
          static void sampleRunReport(String propertyId) throws Exception {
            /**
             * TODO(developer): Uncomment this variable and replace with your
             * Google Analytics 4 property ID before running the sample.
             */
             //propertyId = "269896522";
```

```
            // propertyId = "263363085"

        // Using a default constructor instructs the client to use the credentials
        // specified in GOOGLE_APPLICATION_CREDENTIALS environment variable.
        try (BetaAnalyticsDataClient analyticsData = BetaAnalyticsDataClient.create()) {

          RunReportRequest request1 = RunReportRequest.newBuilder()
              .setProperty("properties/" + propertyId)
//                .addDimensions(
//                            Dimension.newBuilder().setName("country"))
              .addDimensions(
                          Dimension.newBuilder().setName("city"))
              .addMetrics(Metric.newBuilder().setName("activeUsers"))

              .addDateRanges(
                  DateRange.newBuilder().setStartDate("2020-03-31").setEndDate("today")).build();

          RunReportRequest request2 = RunReportRequest.newBuilder()
                  .setProperty("properties/" + propertyId)
              .addDimensions(
                  Dimension.newBuilder().setName("cityId"))
              .addMetrics(Metric.newBuilder().setName("activeUsers"))
              .addDateRanges(
                  DateRange.newBuilder().setStartDate("2020-04-20").setEndDate("today")).build();


          RunReportRequest request3 = RunReportRequest.newBuilder()
                  .setProperty("properties/" + propertyId)
              .addDimensions(
                  Dimension.newBuilder().setName("city"))

             .addMetrics(Metric.newBuilder().setName("activeUsers"))
             .addMetrics(Metric.newBuilder().setName("newUsers"))
             .addMetrics(Metric.newBuilder().setName("totalUsers"))
             .addMetrics(Metric.newBuilder().setName("screenPageViews"))

             .addDateRanges(
                  DateRange.newBuilder().setStartDate("2020-03-31").setEndDate("today")).build();


          RunReportRequest request4 = RunReportRequest.newBuilder()
                  .setProperty("properties/" + propertyId)
              .addDimensions(
                  Dimension.newBuilder().setName("streamName"))

             .addMetrics(Metric.newBuilder().setName("totalUsers"))
             .addMetrics(Metric.newBuilder().setName("screenPageViews"))

             .addDateRanges(
                  DateRange.newBuilder().setStartDate("2020-03-31").setEndDate("today")).build();


          RunReportRequest request5 = RunReportRequest.newBuilder()
                  .setProperty("properties/" + propertyId)
              .addDimensions(
                  Dimension.newBuilder().setName("platform"))
             .addDateRanges(
                  DateRange.newBuilder().setStartDate("2020-03-31").setEndDate("today")).build();

          RunReportRequest request6 = RunReportRequest.newBuilder()
                          .setProperty("properties/" + propertyId)
                      .addDimensions(
                          Dimension.newBuilder().setName("screenResolution"))
                     .addDateRanges(
                          DateRange.newBuilder().setStartDate("2020-03-31").setEndDate("today")).build();

          RunReportRequest request7 = RunReportRequest.newBuilder()
                          .setProperty("properties/" + propertyId)
                      .addDimensions(
                          Dimension.newBuilder().setName("region"))
                     .addDateRanges(
```

```
                    DateRange.newBuilder().setStartDate("2020-03-31").setEndDate("today")).build();

    RunReportRequest request8 = RunReportRequest.newBuilder()
                    .setProperty("properties/" + propertyId)
                .addDimensions(
                    Dimension.newBuilder().setName("pageTitle"))
                .addDateRanges(
                    DateRange.newBuilder().setStartDate("2020-03-31").setEndDate("today")).build();

    RunReportRequest request9 = RunReportRequest.newBuilder()
                    .setProperty("properties/" + propertyId)
                .addDimensions(
                    Dimension.newBuilder().setName("fullPageUrl"))
                .addDateRanges(
                    DateRange.newBuilder().setStartDate("2020-03-31").setEndDate("today")).build();

    RunReportRequest request10 = RunReportRequest.newBuilder()
                    .setProperty("properties/" + propertyId)
                .addDimensions(
                    Dimension.newBuilder().setName("eventName"))
                .addMetrics(Metric.newBuilder().setName("eventCount"))
                .addMetrics(Metric.newBuilder().setName("eventValue"))
                .addMetrics(Metric.newBuilder().setName("eventCountPerUser"))
                //.addMetrics(Metric.newBuilder().setName("eventLabel"))
                .addDateRanges(
                    DateRange.newBuilder().setStartDate("2020-03-31").setEndDate("today")).build();

   // RunReportRequest request11 = RunReportRequest.newBuilder()
              //      .setProperty("properties/" + propertyId)
        //        .addDimensions(
        //            Dimension.newBuilder().setName("customEvent:Read Concept Code Download"))
        //       .addDateRanges(
        //            DateRange.newBuilder().setStartDate("2020-03-31").setEndDate("today")).build();


    // Make the request.
    RunReportResponse response1 = analyticsData.runReport(request1);
    outputReport(response1);

    RunReportResponse response2 = analyticsData.runReport(request2);
    outputReport(response2);

    RunReportResponse response3 = analyticsData.runReport(request3);
    outputReport(response3);

    RunReportResponse response4 = analyticsData.runReport(request4);
    outputReport(response4);

    RunReportResponse response5 = analyticsData.runReport(request5);
    outputReport(response5);

    RunReportResponse response6 = analyticsData.runReport(request6);
    outputReport(response6);

    RunReportResponse response7 = analyticsData.runReport(request7);
    outputReport(response7);

    RunReportResponse response8 = analyticsData.runReport(request8);
    outputReport(response8);

    RunReportResponse response9 = analyticsData.runReport(request9);
    outputReport(response9);

    RunReportResponse response10 = analyticsData.runReport(request10);
    outputReport(response10);

    //RunReportResponse response11 = analyticsData.runReport(request11);
    //outputReport(response11);
  }
}
```

```
  public static void outputReport(RunReportResponse response) {
         System.out.println("\nReport result:\n");

         // Iterate through dimension headers from response.
         List<DimensionHeader> headerList = response.getDimensionHeadersList();
         Iterator<DimensionHeader> headerIterator = headerList.iterator();
         while (headerIterator.hasNext()) {
    System.out.print(headerIterator.next().getName() + "  ##  ");
         }

         // Iterate through dimension headers from response.
         List<MetricHeader> headerListMetrics = response.getMetricHeadersList();
         Iterator<MetricHeader> headerIteratorMetrics = headerListMetrics.iterator();
         while (headerIteratorMetrics.hasNext()) {
    System.out.print(headerIteratorMetrics.next().getName() + "  ##  ");
         }

         System.out.println("");

         // Iterate through values for response.
         for (Row row : response.getRowsList()) {

                 for (int x = 0; x < row.getDimensionValuesCount(); x++) {
                        System.out.print(row.getDimensionValues(x).getValue()+ "  ##  ");
                 }
                 for (int y=0; y < row.getMetricValuesCount(); y++) {
                         System.out.print(row.getMetricValues(y).getValue()+ "  ##  ");
                 }
      }
         System.out.println("\n-----");
  }
}
```

Sample Output

Sample Output

Report result:

city ## activeUsers ##
Bethesda ## 8 ##

Report result:

cityId ## activeUsers ##
1018516 ## 8 ##

Report result:

city ## activeUsers ## newUsers ## totalUsers ## screenPageViews ##
Bethesda ## 8 ## 8 ## 8 ## 236 ##

Report result:

streamName ## totalUsers ## screenPageViews ##
EVS Report Exporter PROD ## 8 ## 236 ##

Report result:

platform ##
web ##

Report result:

screenResolution ##
1280x720 ## 1536x864 ## 1920x1080 ## 1920x1200 ##

Report result:

region ##
Maryland ##

Report result:

pageTitle ##
EVS Report Exporter - Branch Resolve ## EVS Report Exporter - Documentation ## EVS Report Exporter - Downloads ## EVS Report Exporter - Read Code ## EVS Report Exporter - Report Selection ##

Report result:

fullPageUrl ##
evs.cancer.gov/report-exporter/ ## evs.cancer.gov/report-exporter/documentation ## evs.cancer.gov/report-exporter/exports ## evs.cancer.gov/report-exporter/readCodeEntry ## evs.cancer.gov/report-exporter/resolveBranchEntry ##

Report result:

eventName ## eventCount ## eventValue ## eventCountPerUser ##
page_view ## 264 ## 0 ## 33 ## scroll ## 66 ## 0 ## 8.25 ## click ## 40 ## 0 ## 13.333333333333334 ## Read Concept Code Download ## 30 ## 0 ## 7.5 ## user_engagement ## 23 ## 0 ## 3.2857142857142856 ## session_start ## 14 ## 0 ## 1.75 ## Branch Resolve Deferred Download ## 11 ## 0 ## 3.6666666666666665 ## Deferred Download ## 11 ## 0 ## 3.6666666666666665 ## Branch Resolve Download ## 9 ## 0 ## 3 ## first_visit ## 8 ## 0 ## 1 ##