Using WebSphere Cast Iron with OAuth to access Google APIs and Amazon Redshift, Part 1: Extracting data from Google Analytics

Giovanni Vigorelli (giovanni@nz.ibm.com)
IT Specialist
IBM

Jervis Lee (jervisl@tw.ibm.com)
IT Specialist
IBM

This two-part tutorial series demonstrates the flexibility of WebSphere® Cast Iron by integrating with popular cloud services such as Google® and Amazon® Web Services. The tutorial series shows how to authenticate to Google Analytics via Google APIs and OAuth V2, and how to use a JDBC connection to copy data to the Amazon Redshift database. Part 1 describes how to configure Cast Iron endpoints to integrate with Google APIs and Amazon Redshift.

View more content in this series

Introduction

In the last few years, most organizations started leveraging cloud services as an incubator of new projects. Once they realized that the cloud was not just a sandpit, but that it could offer reliable and secure services, the adoption of cloud as an extension of on-premise applications grew exponentially.

While there are a variety of services that can cover most of these needs (CRM, billing, storage, messaging, ticketing, to name a few), organizations still have the challenge to integrate data that is spread in several places to have a single view. WebSphere Cast Iron Integration (hereafter called Cast Iron) is a tool "born to be a cloud". By nature, it offers capabilities to integrate cloud service to cloud service and cloud service to on-premise service. Those capabilities allow business to overcome the boundaries of IT infrastructure and access data as a single pane of glass.

This tutorial is divided in two parts. In Part 1, we cover the configuration of Google API (OAuth and Analytics), Amazon Redshift, and endpoints in Cast Iron to use these technologies. In Part 2, we will build an orchestration to have an end-to-end working workflow. We will illustrate how to gather data from Google Analytics and store that into a relational database (Amazon Redshift). This
This tutorial can help you extend the above mentioned scenario to other cloud services (for example, SalesForce®.com, Chargify®.com, and Oracle® CRM OD) and provide a single source of truth in the target database (Redshift in this scenario).

This tutorial assumes that you have some familiarity with WebSphere Cast Iron Studio (development tool) and basic knowledge of technology such as HTTP, SQL, and JavaScript.

**Figure 1. Tutorial scenario**

![Tutorial Scenario Diagram](image)

**Setting up Google Analytics and APIs**

This section covers at a high level what you will be getting from Google Analytics and, most importantly, how to set up a Google account to permit third party applications, such as Cast Iron Integration, to access Google APIs using OAuth V2.

1. Go to the [Google Developer Console](https://console.developers.google.com) and enable Google APIs as shown in Figure 2.

**Figure 2. Enable Google APIs**

![Google APIs Enablement](image)

2. Configure Google Analytics and your application to start gathering data as shown in Figure 3 and Figure 4.
Figure 3. Add an API tracking ID to the application

Figure 4. Add API tracking ID to the application

3. After a few hours, check the Google Analytics dashboard to see if any data has been connected as shown in Figure 5.
Figure 5. Google Analytics report

4. At this point, you are ready to explore the world of OAuth V2 in Google API. Google requires that any access to API is done via OAuth V2, which is "slightly" more sophisticated and complex than basic authentication that is used by other cloud applications. Go to the Google Developer Console. Once you have created a project under APIs & auth – Credentials, select Create new Client ID as shown in Figure 6.

Figure 6. Create an API Client ID

5. In the configuration panel, select Installed application and Other as shown in Figure 7.
A new entry with CLIENT ID, CLIENT SECRET, and REDIRECT URIS is created as shown in Figure 8.

**Figure 8. API security details**

6. In the same console, configure the "APIs & auth – Credentials consent" screen as shown in Figure 9. The email and product name are required, but you can decide what you want to use.
7. You need to grant access to Google Analytics for the given Client ID. Send a request to https://accounts.google.com/o/oauth2/auth. This is a simple HTTP GET call so it can be done with any HTTP client like a browser. Some required parameters have to be passed as Query parameters as shown below:

```text
scope=https://www.googleapis.com/auth/analytics
redirect_uri=urn:ietf:wg:oauth:2.0:oob
response_type=code
client_id=927890000889-sxxxxx.apps.googleusercontent.com
```

`client_id` comes from Step 5 (above), `redirect_uri` is the default value, and `response_type` is the default value.

You can find scope in the Google API documentation, or you can use the Google OAuth Playground. You do not have to complete the whole procedure, but you can see the construction of the request from Step 1. The URL to call is similar to the one shown below:

```text
https://accounts.google.com/o/oauth2/auth?
response_type=code&client_id=927890000889-sbxxxxxxxxxxpdhk6u.apps.googleusercontent.com
```

Before the request is completed, a consent screen appears (Figure 10) where you need to grant access to the API.
Figure 10. Consent screen

Once accepted, a response with a code is returned as shown in Figure 11.

Figure 11. Response code

8. Once accepted, a response with a code is returned as shown in Figure 11.

Copy the code and save it for later use. Note that the code is longer than the section where it is shown: 4/-IkmoYhxxxxxxxxxi0lcVgrKXntQAax28tg3QkQI

9. The last step to get the OAuth tokens (access and refresh) is to make an HTTP POST call to the Google OAuth 2.0 Authorization server (https://accounts.google.com/o/OAuth2/token). You need to pass the following parameters:

   code = 4/-IkmoYhxxxxxxxxxi0lcVgrKXntQAax28tg3QkQI
   client_id = 92789000xxxxxxespdhk6u.apps.googleusercontent.com
   client_secret = PF6jhxxxxxVvLn4b
   redirect_uri = urn:ietf:wg:OAuth:2.0:oob
   grant_type = authorization_code

   code comes from Step 8 above. client_id and client_secret comes from Step 5. In this tutorial, Postman (a Chrom app) has been used (Figure 12), but you can use any other HTTP client to submit the POST call.
Figure 12. REST client

Here's a sample response:

```json
{
  access_token: "ya29.lwAadj2jbxxxxx...67UyVESn008B-pVte6k5",
  token_type: "Bearer",
  expires_in: 3600,
  refresh_token: "1/9AghLERxxxxxXG7p8ZF-wnkJL0PZFc"
}
```

Save the response data in a secure place. That data will be used shortly to configure Cast Iron.

10. Before moving to Cast Iron, let's build the query string to get data from Google Analytics. Google provides a handy tool called the Query Explorer (Figure 13) to graphically build the query string.
Figure 13. Query Explorer

Google Analytics Query Explorer 2

Sometimes you just need to explore. This tool lets you play with the Core Reporting API by building queries to get data from your Google Analytics views (profiles). You can use these queries in any of the client libraries to build your own tools.

Click on Get Data to check whether there is available data. Click on the world icon button to get the URL to be used later in Cast Iron. Here is a sample URL:


The URL is encoded. There are several tools to decode that, such as the URL Decoder/Encoder. Here’s a decoded sample URL:


Note: In the Cast Iron section, it is important to make the same selection as described above. The data will be transformed and the transformation will work only if the source data has the expected format.

Setting up Amazon Redshift

Amazon Redshift (or AWS Redshift) is a relational database available as Software as a Service (SaaS) on Amazon Web Services. IBM® has alternative solutions for cloud storage, such as Cloudant NoSQL DB or SQL Database available in IBM Bluemix™. To demonstrate the openness of Cast Iron Integration, Redshift is used instead.
This tutorial does not cover all the aspects of creating an instance of Redshift on Amazon. We assume that it is already available and you have the information to access the database instance.

1. Open the AWS Console and open the Redshift console as shown in Figure 14.

**Figure 14. AWS Console**

![AWS Console](image)

1. From the list of clusters, click on the target cluster. In this tutorial, it is **my-dw-instance** as shown in Figure 15. Table 1 shows the values.

**Figure 15. Redshift cluster**

![Redshift cluster](image)

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Cluster Status</th>
<th>DB Health</th>
<th>In Maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>my-dw-instance</td>
<td>available</td>
<td>healthy</td>
<td>no</td>
</tr>
</tbody>
</table>

3. In the cluster configuration section, all the information required to connect from a SQL client is listed as shown in Figure 16.
Figure 16. Instance details

Table 1. Configuration information

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endpoint</td>
<td>my-dw-instance.chgkaitovc2r.us-west-2.redshift.amazonaws.com</td>
</tr>
<tr>
<td>Port</td>
<td>5439</td>
</tr>
<tr>
<td>DB name</td>
<td>sample</td>
</tr>
<tr>
<td>Username</td>
<td>awsuser</td>
</tr>
</tbody>
</table>

4. AWS Redshift does not provide any graphical user interface to interact with the database. It is possible to use any SQL client, which supports JDBC connectivity. In this tutorial, SQuirreL has been used and Amazon recommends SQL Workbench. However, it is a matter of personal choice.

The SQuirreL setup is not described in this tutorial, but you need to know that AWS Redshift supports PostgreSQL for JDBC connectivity. You need to get the JDBC Driver to configure both SQuirreL and Cast Iron Integration later on. You can download the driver from the PostgreSQL web site. We will use postgresql-9.3-1102.jdbc41.jar in this tutorial.

5. Once the JDBC driver has been imported in SQuirreL, configure a database connection (“alias” on the tool). This is useful to create the table that will be used in Cast Iron later on and to verify that everything is working. See Figure 17.
6. Once connected, create the target table in the database (Figure 18). A simple table is shown below:

```sql
create table googleAnalyticsData(
sessions integer,
bounces integer,
city varchar(30),
browser varchar(30));
```

You can explore the database object as well to check that everything has been correctly created as shown in Figure 19.
Figure 19. Data check

7. As an additional test, you can insert and retrieve records into the table as shown in Figure 20 and Figure 21.

```sql
INSERT INTO googleAnalyticsData (SESSIONS, BOUNCES, CITY, BROWSER)
VALUES (5, 4, 'New York', 'Firefox');
SELECT * FROM googleAnalyticsData
```

Figure 20. Insert data
Figure 21. Retrieve data

![Image of database query]

To clean the table, enter `DELETE FROM googleAnalyticsData`.

You now have all the components and information to start working on Cast Iron Integration.

**Defining a Cast Iron – Google Analytics endpoint**

Cast Iron Integration provides a development tool called Studio (or Designer Web Start if working with Cast Iron Live). This tutorial assumes that you are familiar with the tool and not all the tasks will have step-by-step instructions.

The first step is to create an HTTP endpoint that gives access to Google API. As mentioned above, OAuth V2 will be used as the authorization mechanism to get data from Google Analytics. Keep handy all the information gathered in the previous section. The configuration is straightforward once you have that data.

1. Open **Cast Iron Studio**, create a new project, and add a new **HTTP Endpoint** as shown in Figure 22. Give it a meaningful way such as `HTTP-GoogleAPI`. Now it is time to fill the endpoint configuration in the Object Editor. Figure 22 shows all the parameters that need to be filled. At the bottom of Figure 22, you see a list of parameters with the value gathered in the **Setting up Google Analytics and APIs** section. Table 2 describes the items shown in Figure 22.
Using WebSphere Cast Iron with OAuth to access Google APIs and Amazon Redshift, Part 1: Extracting data from Google Analytics

Figure 22. HTTP endpoint configuration

Table 2. HTTP endpoint parameters

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td><a href="http://www.googleapis.com443">www.googleapis.com443</a></td>
<td>Google Analytics destination address</td>
</tr>
<tr>
<td>B</td>
<td>authorization_code</td>
<td>Setting up Google Analytics and APIs</td>
</tr>
<tr>
<td>C</td>
<td><a href="https://www.googleapis.com/auth/analytics">https://www.googleapis.com/auth/analytics</a></td>
<td>Setting up Google Analytics and APIs</td>
</tr>
<tr>
<td>D</td>
<td>92789000...</td>
<td>Setting up Google Analytics and APIs</td>
</tr>
<tr>
<td>E</td>
<td>PF6jFvpJp3e...</td>
<td>Setting up Google Analytics and APIs</td>
</tr>
<tr>
<td>F</td>
<td>ya29.1wAadj...</td>
<td>Setting up Google Analytics and APIs</td>
</tr>
<tr>
<td>G</td>
<td><a href="https://accounts.google.com/o/OAuth2/token">https://accounts.google.com/o/OAuth2/token</a></td>
<td>Google API authentication URL</td>
</tr>
<tr>
<td>H</td>
<td>1/9AghLERQ...</td>
<td>Setting up Google Analytics and APIs</td>
</tr>
</tbody>
</table>
2. You can now test what you have done so far. Scroll down at the bottom of the HTTP Endpoint configuration to see the **Test Connection** button. Before clicking the button, copy the target URL to the clipboard. This value has been defined in Setting up Google Analytics and APIs:

3. Click the **Test Connection** button as shown in Figure 23.

**Figure 23. Test connection**

Fill the **Server Resources** and confirm. A successful connection message appears as shown in Figure 24.
Defining the Cast Iron – AWS Redshift endpoint

This section repeats conceptually what has been done in the previous one, which is the way Cast Iron Integration communicates with remote services.

While the functionality is the same (provide connectivity), the implementation is completely different (HTTP versus JDBC), thus the endpoint definition has different parameters.

1. First, you need to import the JDBC driver in Studio that is used to establish a connection and a dialog with AWS RedShift. Studio uses the same JAR file used to configure SQuirreL (postgresql-9.3-1102.jdbc41.jar). In Studio, click on Tools > Install Module Providers as shown in Figure 25.

Figure 25. Module providers

Click on the plus sign under Database Connector, then Browse, and select the postgresql-9.3-1102.jdbc41.jar file from the file system as shown in Figure 26.
2. In the same project created in the previous section, add a new **Database Endpoint** and give it a meaningful name, such as **DB-Redshift**. Fill the values in the endpoint editor as shown in Figure 27. Those parameters have been identified in **Setting up Amazon Redshift**.
Figure 27. JDBC endpoint configuration

![JDBC endpoint configuration figure]

Table 3 describes the items shown in Figure 27.

### Table 3. JDBC endpoint parameters

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Generic JDBC Driver</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>org.postgresql.jdbc2.optional.SimpleDataSource</td>
<td>Click on T symbol to get a drop down list of available classes.</td>
</tr>
<tr>
<td>C</td>
<td>sample</td>
<td>The name you selected</td>
</tr>
<tr>
<td>D</td>
<td>my-dw-instance.chgkaitovc2r.us-west-2.redshift.amazonaws.com</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>5439</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>awsuser</td>
<td>The name you selected</td>
</tr>
<tr>
<td>G</td>
<td>&lt;password&gt;</td>
<td>The name you selected</td>
</tr>
</tbody>
</table>

3. Scroll down to the bottom of the Endpoint editor to see the **Test Connection** button and click it. You now see a successful connection window as shown in Figure 28.
Troubleshooting tips

With Cast Iron Studio, you can check for error messages and variable values of each activity of the orchestration in the "Variable/Parameter Message Data" section after you verify the orchestration.

Another source of information is the "error.log" file in the Studio installation directory, such as $C:\Program Files\IBM\WebSphere Cast Iron Studio 7.0.0.1$.

If Designer WebStart (Studio from Cast Iron Live) is used instead, the "error().log" file is located in the .castiron directory in the user’s home directory, such as $C:\Users\Administrator\.castiron$.

Conclusion

In Part 1 of this 2-part series, you learned how to configure all the components, Google APIs, Amazon Redshift, and Cast Iron Integration endpoints. You also learned how to create an integration flow and orchestration. In Part 2, you will create a sample orchestration to save the data from Google Analytics into a database table in Amazon Redshift.
Resources

- IBM WebSphere Cast Iron V7 documentation
- Google Analytics
- Google API
- Amazon Redshift
- SQL Client
- URL Decoder/Encoder
- developerWorks WebSphere resource page
About the authors

Giovanni Vigorelli

Giovanni Vigorelli is currently a Pre-sales Specialist in the middleware brand. He works with clients as a technical specialist. Prior to that, he worked as a Subject Matter Expert in WebSphere by providing support to clients.

Jervis Lee

Jervis Lee is a WebSphere Cast Iron L2 Support Engineer at the IBM Taiwan Software Development Lab. He has worked as a technical leader of the WebSphere Transformation Extender Performance Testing team. His expertise covers performance testing, web application, development, and information security.

© Copyright IBM Corporation 2015
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)