HCLSoftware

Unica Link V12.1.8 Google Connector User Guide

Contents

Chapter 1. Overview

The Google Ads connector uploads contact information into an existing/new audience in Google Ads.

Chapter 2. Prerequisites

To configure Google Ad Connector in Unica, you must have:

- a Google Ad Manager account
- a Google Developer account

Table 1. Accessing Credentials

Credential Type	How to access?
Google Developer Account Client ID	Get this from Google developer account.
Google Developer Account Client Secret	
Google Ad Manager Developer Token	Get this from Google Ad Manager account.
Google Ad Manager Customer ID	
Google Ad Account ID	Get this from Google Ad account.

Accessing details from Google Developer account

To access details from the Google Developer account, via API, you need a Client ID and Client Secret.

About this task

If you do not have a Client ID or Client Secret, use your Google Developer account to generate the Client ID and Client Secret.

To generate the Client ID and Client Secret, complete the following steps:

1. Log in to https://console.cloud.google.com/.
2. Select the project that you want to use for Ad integration.
3. Enable **Google Ads API** within the selected project.
4. Select **Credentials**.
5. Under **OAuth 2.0 Client IDs**, generate the credentials using the Desktop app.
   Result
   A screen opens containing the Client ID and Client Secret.

What to do next

Ensure that:

- OAuth consents are configured
- the app is published
- the concerned user has the required access

Accessing details from Google Ad Manager account

To access the Google Ad Manager Developer token, complete the following steps:

1. Log in to your Google Ad Manager account.

2. Navigate to **Tools and Settings > API Centre**

   Result

   The **API Centre** page opens.

3. Create a new **Developer token**. For a new **Developer token**, you must provide the following details:

   - API contact email
   - Company name
   - Company URL
   - Company type
   - Intended use, and other details

4. By default, the **Developer token** is generated for use with the `Test Account`. The generated **Developer token** will not work with the `Production Account`. If you want the **Developer token** to work with the `Production Account`, you must **Apply for Basic Access**.

   Note: Getting a Basic level access or a Standard level access from Google for the generated **Developer token** is a time-consuming process. We recommend that you initiate the review process with Google at the earliest.

Chapter 3. Configuration

To use Google Ads connector, Unica administrator must configure a connection.

1. To automatically generate the Authorization URL, provide the Client Id and the Client Secret.
2. Copy-and-paste the URL on a browser and sign-in with your Google Ads Account.
3. To automatically retrieve the Refresh Token, copy-and-paste the Authorization code from the browser to Google Ads connection properties.
4. The input fields to be provided are as follows:
     - **Developer Token**
     - **Customer Id**

The connection properties are mentioned in the following table:

Properties	Description
Google Developer Account Client ID	Google Developer Account Client ID
Google Developer Account Client Secret	Google Developer Account Client Secret
Authorization URL	Authorization URL
Authorization Code	Authorization Code
Refresh Token	Refresh Token
Google Ad Manager Developer Token	Google Ad Manager Developer Token
Google Ad Manager Customer ID	Google Ad Manager Customer ID

Chapter 4. Test Connection

Perform the following steps to test a connection: When defining a connection, the Test button can be pressed. This invokes a query operation on Google Ads to ensure that server can be reached and that the connection parameters are valid.

When defining a connection, click the **Test** button.

The system invokes a query operation on Google Ads to confirm that:

- the server is reachable.
- the connection parameters are valid.

Chapter 5. Touchpoint or Process Box Configuration

When you configure Google Ads connector from a Touchpoint in Unica Journey or a Process box in Unica Campaign, the following properties are displayed:

Property	Description
Google Ad Account ID	Google Ad Account ID.
Mode	The possible options in the dropdown are Create User List and Append User List.
User List Name	This field will be enabled if the mode is equal to Create User List.
User List Description	This field will be enabled if the mode is equal to Create User List.
Membership Life Span	(User lists can use a membership life span of 10,000 to indicate unlimited). This field will be enabled if the mode is equal to Create User List.
User List	This field will be enabled if the mode is equal to Append User List.

The field mapping screen would be generated based on selected upload type and for the details related to data normalization, see Data Normalization on page 8. The fields available for Upload Users are as follows:

Field	Description
First Name	The first name of the user data.
Last Name	The last name of the user data.
City	The city of the user.
State	The state of the user.
Country Code	The two-letter country code.
Postal Code	The postal code of the user.
Email	The email address.
Phone Number	The phone number of the user.

Chapter 6. Updating `config.yaml` for Proxy Server

This topic describes the updates in `config.yaml` file for using proxy server for external communication.

About this task

To update the properties in `config.yaml` for using proxy server for external communication, complete the following steps:

1. Update the existing lines in server section of `config.yaml` for the long and short workers as shown here:

   ```
   longTaskProcessJvmOptions: "-Xmx2g -Dhttps.proxyHost=<HOST> -Dhttps.proxyPort=<PORT>"
   shortTaskProcessJvmOptions: "-Dhttps.proxyHost=<HOST> -Dhttps.proxyPort=<PORT>"
   ```

   where:

   - *<HOST>* is the IP address or domain name of the proxy server. For example: `Proxy IP/FQDN/Hostname`.
   - *<PORT>* is the Port number of the proxy server.

2. Go to "JVM Options" under "runtime" section of `config.yaml` and update the following option for JVM as shown here:

   ```
   option4: "-Dhttps.proxyHost=<HOST> -Dhttps.proxyPort=<PORT>"
   ```

   where:

   - *<HOST>* is the IP address or domain name of the proxy server. For example: `Proxy IP/FQDN/Hostname`.
   - *<PORT>* is the Port number of the proxy server.

Chapter 7. Data Normalization

This topic describes the formatting guidelines for uploading hashed data for Google connector.

The following table list the data format layout information for Google connector:

Field	Description	Remarks
`First Name`	The first name of the user data.	Do not include prefixes. For example: `Mrs.`
`Last Name`	The last name of the user data.	Do not include suffixes. For example: `Jr.`
`City`	The city of the user.	
`State`	The state of the user.	
`Country Code`	The two-letter country code.	Use ISO two-letter or three-letter country codes and include the country code even if all your customer data is from the same country. For example: • `US` or `USA` for United States of America. • `SG` or `SGP` for Singapore
`Zip or Postal Code`	The zip or postal code of the user.	Both US and international zip and postal codes are allowed. For United States: • 5-digit codes are allowed (for example: `94303`). • 5 digits followed by 4-digit extension are also allowed and may improve your match rate (for example: `94303-5300`). For all other countries: • Leave out postal code extensions (for example: `WC2H 8LG`).
`Email`	The email address.	Include a domain name for all email addresses (for example, `gmail.com` or `hotmail.co.jp`).

Field	Description	Remarks
		Remove any spaces in between the email address. For example: • `example@email.com` • `test@gmail.com`
`Phone Number`	The phone number of the user.	Include the country code. For example: • `1 (234) 567-8910` • `81-12-3456-7891` • `02 1234 5678`

Note:

- Link connector is using following fields for Google and not performing any data normalization including data hashing at the connector level:
  - FirstName
  - LastName
  - City
  - State
  - CountryCode
  - Zip or PostalCode
  - Email
  - PhoneNumber
- Google Ads is taking care of data normalization including data hashing with SHA256 by using the Google SDK method normalizeAndHash for data normalization on following fields:
  - Email
  - PhoneNumber
  - FirstName
  - LastName

Chapter 8. Event Tracking

The responses that you receive from the Google Ads connector are as follows:

- `Audience_id`
- `Session_id`
- `Num_received`
- `Num_invalid_entries`
- `Num_valid_entries`
- `status`
- `errormessage`
- `timestamp`

Example

Success Scenario:

```
xxxx,,2,2,2,Success,,2021-07-22T13:20:38
```

Error Scenario:

```
,,2,0,0,error,Invalid cu,2021-07-22T13:22:35
```

Chapter 9. Google API/SDK details

HCL Unica Link uses the Google Ad API to perform the following activities:

- Connect to the google ad platform and authorize the user.
- List the "audiences" (user lists).
- Create new audience list (user list).
- Upload/Append users to the user list and API response is sent back to Unica.

Note: The Google ad API user lists are referred as "Audiences" in the HCL Unica application.

Host name and port details

- Host name for Google Ad API is `googleads.googleapis.com` and `oauth2.googleapis.com`.
- Port number for Google Ad API is 443.

Packages Imported and API Services/Methods Called

```
import com.google.ads.googleads.lib.GoogleAdsClient;
import com.google.ads.googleads.<Google API version>.common.CrmBasedUserListInfo;
import com.google.ads.googleads.<Google API version>.common.CustomerMatchUserListMetadata;
import com.google.ads.googleads.<Google API version>.common.OfflineUserAddressInfo;
import com.google.ads.googleads.<Google API version>.common.UserData;
import com.google.ads.googleads.<Google API version>.common.UserIdentifier;
import com.google.ads.googleads.<Google API
 version>.enums.CustomerMatchUploadKeyTypeEnum.CustomerMatchUploadKeyType;
import com.google.ads.googleads.<Google API version>.enums.OfflineUserDataJobTypeEnum.OfflineUserDataJobType;
import com.google.ads.googleads.<Google API version>.resources.OfflineUserDataJob;
import com.google.ads.googleads.<Google API version>.resources.UserList;
import com.google.ads.googleads.<Google API version>.services.AddOfflineUserDataJobOperationsRequest;
import com.google.ads.googleads.<Google API version>.services.AddOfflineUserDataJobOperationsResponse;
import com.google.ads.googleads.<Google API version>.services.CreateOfflineUserDataJobResponse;
import com.google.ads.googleads.<Google API version>.services.GoogleAdsRow;
import com.google.ads.googleads.<Google API version>.services.GoogleAdsServiceClient;
import com.google.ads.googleads.<Google API version>.services.MutateUserListsResponse;
import com.google.ads.googleads.<Google API version>.services.OfflineUserDataJobOperation;
import com.google.ads.googleads.<Google API version>.services.OfflineUserDataJobServiceClient;
import com.google.ads.googleads.<Google API version>.services.SearchGoogleAdsStreamRequest;
import com.google.ads.googleads.<Google API version>.services.SearchGoogleAdsStreamResponse;
import com.google.ads.googleads.<Google API version>.services.UserListOperation;
import com.google.ads.googleads.<Google API version>.services.UserListServiceClient;
import com.google.api.gax.rpc.ServerStream;
import com.google.auth.oauth2.ClientId;
import com.google.auth.oauth2.UserAuthorizer;
import com.google.auth.oauth2.UserCredentials;
import com.google.common.collect.ImmutableList;
```

User Authorization through HCL Unica

```
com.google.auth.oauth2.UserAuthorizer::build
com.google.auth.oauth2.UserAuthorizer::getAuthorizationUrl
com.google.auth.oauth2.UserAuthorizer::getCredentialsFromCode
```

```
com.google.auth.oauth2.UserCredentials::getRefreshToken
```

Get User List

```
com.google.ads.googleads.<Google API version>.services.GoogleAdsServiceClient::searchStreamCallable
com.google.ads.googleads.<Google API version>.services.SearchGoogleAdsStreamRequest::build
com.google.ads.googleads.<Google API version>.services.SearchGoogleAdsStreamResponse::getResultsList
com.google.api.gax.rpc.ServerStream::iterator
com.google.ads.googleads.<Google API version>.services.GoogleAdsRow::getUserList
```

Create a Customer Match user list

```
com.google.ads.googleads.<Google API version>.resources.UserList::build
com.google.ads.googleads.<Google API version>.resources.UserListOperation::build
com.google.ads.googleads.lib.GoogleAdsClient::createUserListServiceClient
com.google.ads.googleads.<Google API version>.resources.UserListServiceClient::mutateUserLists
```

Create and execute an asynchronous job to add users to the Customer Match user list

```
com.google.ads.googleads.lib.GoogleAdsClient::createOfflineUserDataJobServiceClient
com.google.ads.googleads.<Google API version>.services.OfflineUserDataJobServiceClient::build
com.google.ads.googleads.<Google API version>.services.OfflineUserDataJobServiceClient::createOfflineUser
 DataJob
com.google.ads.googleads.<Google API version>.services.OfflineUserDataJobServiceClient::addOfflineUserD
 ataJobOperations
com.google.ads.googleads.<Google API version>.services.OfflineUserDataJobServiceClient::addAllOperations
com.google.ads.googleads.<Google API version>.services.OfflineUserDataJobServiceClient::setEnablePartial
 Failure
com.google.ads.googleads.<Google API version>.resources.OfflineUserDataJob::build
 com.google.ads.googleads.v12.services.CreateOfflineUserDataJobResponse::getResource Name
com.google.ads.googleads.<Google API
 version>.services.AddOfflineUserDataJobOperationsResponse::hasPartialFailureError
 com.google.ads.googleads.<Google API version>.services.OfflineUserDataJobOperation::build
```

12