# Google Ads Import Integration - V2

CLUse this data connector to import Google Ads reports into Treasure Data.

This topic includes:



## Prerequisites

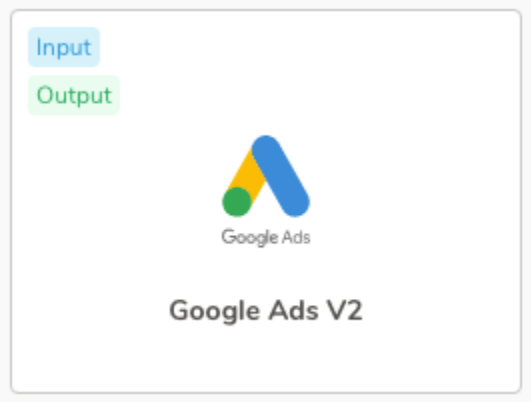
- Basic knowledge of Treasure Data
- Basic knowledge of Google Ads

## Requirements and Limitations

- Using searchStream could fail the job if the data is too big. Using paging could fix that problem.

## Import from via TD Console

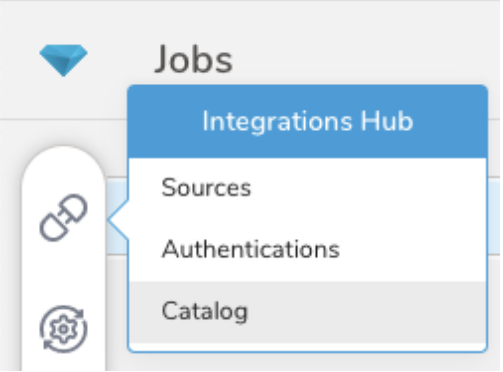
Navigate to Integrations Hub to Catalog, search and select Google Ads input. The following dialog opens.




### Create Authentication

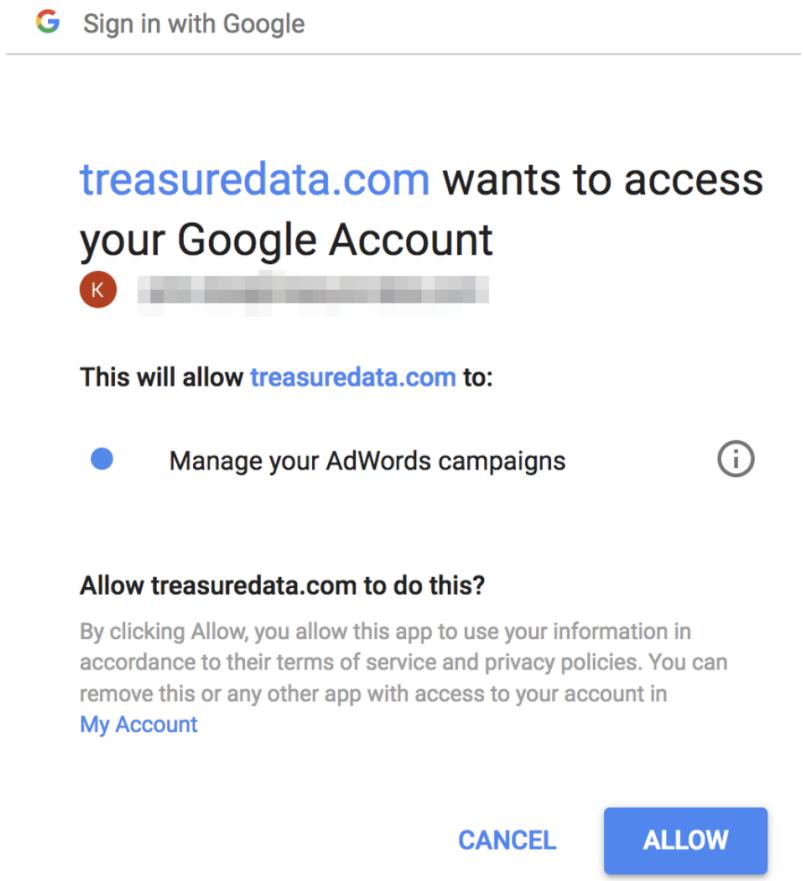
1. Select **Integrations Hub**.

2. Select **Catalog**.

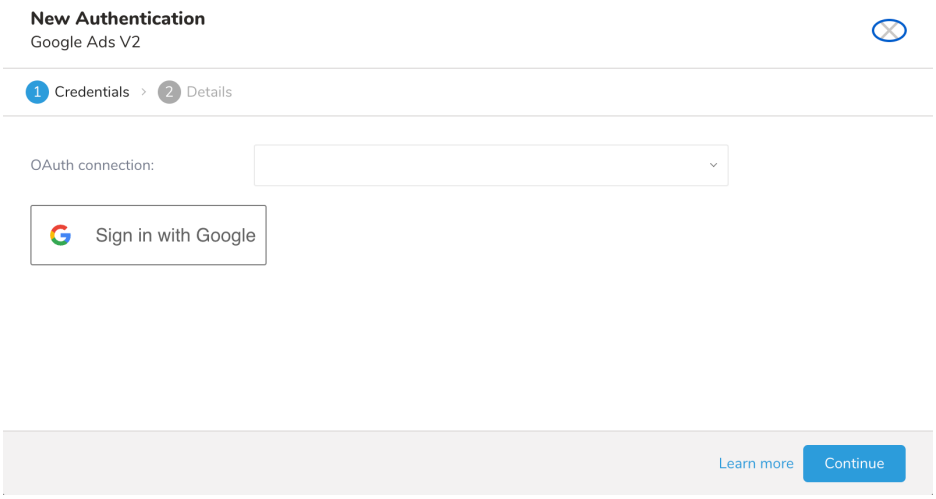



3. Search for your Integration in the Catalog; hover your mouse over the icon and select **Create Authentication**.


Ensure that the **Credentials** tab is selected and then select it to connect a new account.
Log into your Google Ads account from the new window and grant `Treasure Data` access to your **Ads campaigns**:




You are redirected back to Integrations Hub to Catalog. Repeat the step to connect to a new account to choose your new OAuth connection.



4. Enter a name for your authentication, and select **Done**.

## Create a Source

1. Open TD Console.
2. Navigate to **Integrations Hub** > **Authentications**.
3. Locate your new authentication and select **New Source**.

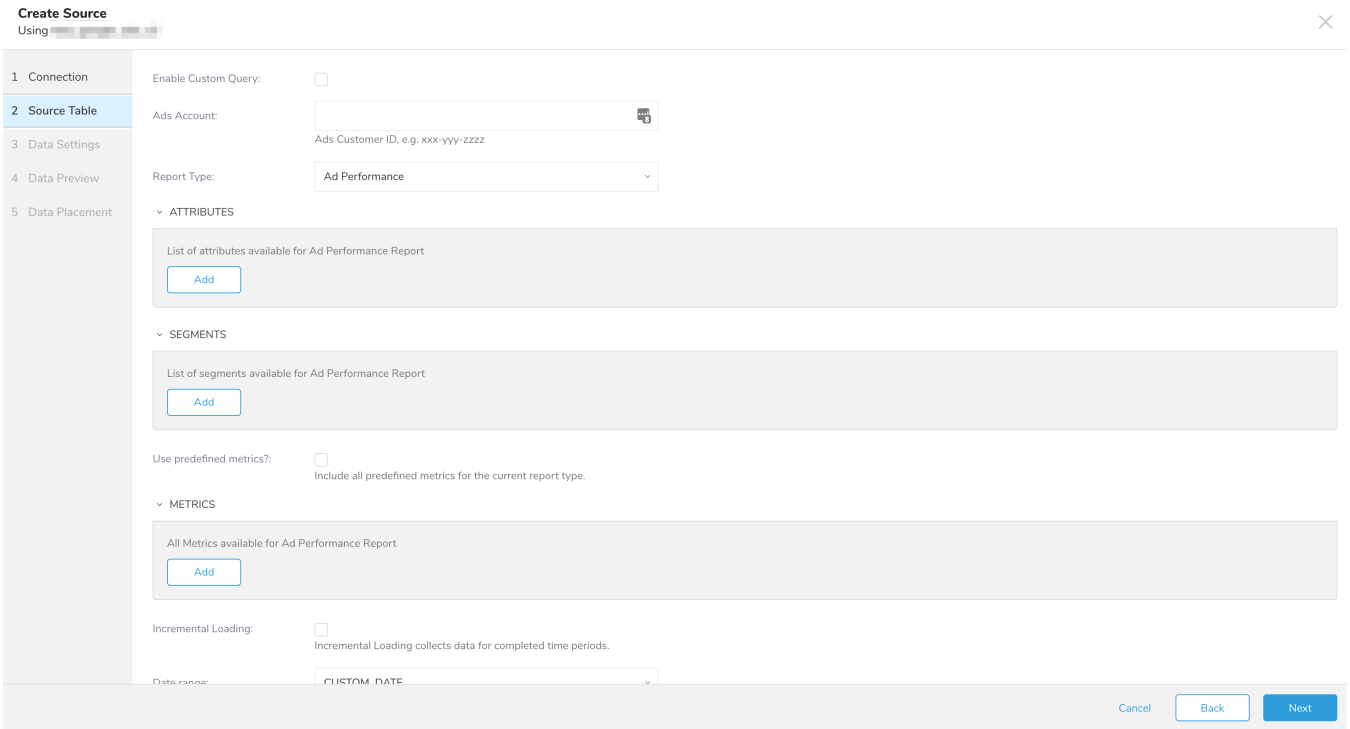
## Create a Connection

| Parameter | Description |
|---|---|
| Data Transfer Name | You can define the name of your transfer. |
| Authentication | The authentication name that is used to a transfer. |

1. Type a source name in the Data Transfer Name field.
2. Select **Next**.

The Create Source page displays with the **Source Table** tab selected.

## Identify a Source Table

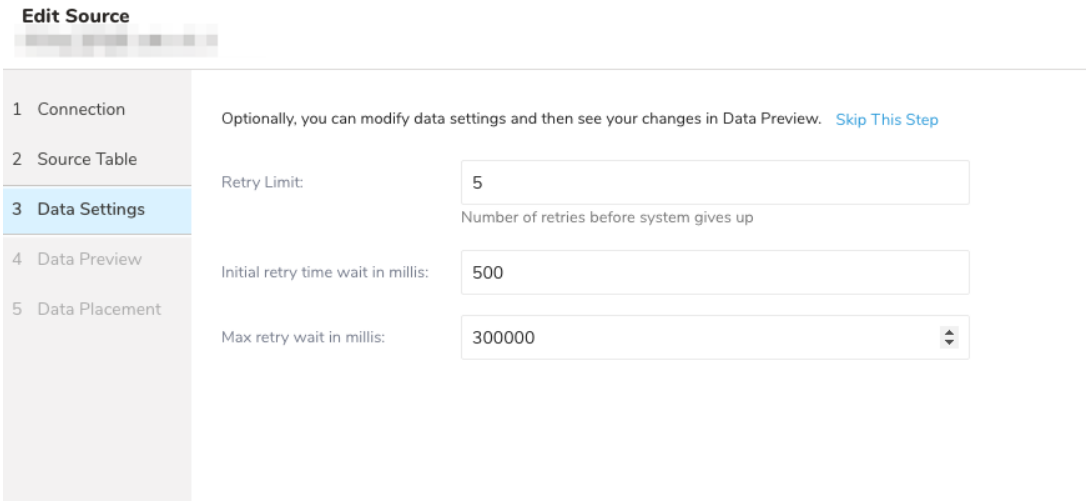



| Parameter | Description |
|---|---|
| client_customer_id | Client customer ID |
| target | Report type |
| segments | List of additional segments |
| metrics | List of additional metrics |
| attributes | List of additional attributes |
| date_range | Date range type |

| | |
|---|---|
| include_zero_impressions | Filter the records by impression |
| include_predefined_metrics | Include all predefined metrics |
| incremental | Run the job in incremental mode |
| start_date | Start date, used with `date_range` is `custom_date` |
| end_date | End date, used with `date_range` is `custom_date` |
| include_negative_keywords | Filter the records by negative keywords, only used with keywords_performance_report |
| enable_custom_query | Enable to use the custom query |
| select_columns | List of fields to query, separated by comma |
| from_target | Report target name |
| other_conditions | The other condition of the query |

Select **Next**.

## Define Data Settings




| Parameter | Description |
|---|---|
| maximum_retries | Internal maximum retries limit |
| initial_retry_interval_millis | init waiting time |
| maximum_retry_interval_milliis | maximum waiting time |

Select **Next**.

## Preview Your Data

You can see a preview of your data before running the import. The data that displays in the data preview is approximated from your source. It is not the actual data that is imported.

1. Select **Next**.
   Data preview is optional and you can safely skip to the next page of the dialog if you want.
2. To preview your data, select **Generate Preview**. Optionally, select **Next**.
3. Verify that the data meets your expectations.
4. Select **Next**.

## Define Your Data Placement

Select the target database and table where you want your data placed, and then indicate how often the import should run.

1. Select **Data Placement.**
2. Select a **Database** > **Select an existing** or **Create New Database**. Optionally, enter a database name.

3. Select a **Table**> **Select an existing** or **Create New Table**. Optionally, type a table name.
4. Choose the method for importing the data.
    - **Append** (default): Data import results are appended to the table.
      If the table does not exist, it will be created.
    - **Always Replace:** Replaces the entire content of an existing table with the resulting output of the query. If the table does not exist, a new table is created.
    - **Replace on New Data:** Only replace the entire content of an existing table with the resulting output when there is new data.
5. Select the **Timestamp-based Partition Key** column.
   If you want to set a different partition key seed than the default key, you can specify the long or timestamp column as the partitioning time. As a default time column, it uses upload_time with the add_time filter.
6. Select the **Timezone** for your data storage.
7. Choose when and how often you want to run this query:
    - **Run once**:
        - Select **Off**.
        - Select **Scheduling Timezone**.
        - Select **Create & Run Now**.
    - **Repeat the query**:
        - Select **On**.
        - Select the **Schedule**. The UI provides these four options: *@hourly*, *@daily,* and *@monthly* or custom *cron*.
        - You can also select **Delay Transfer** and add a delay of execution time.
        - Select **Scheduling Timezone**.
        - Select **Create & Run Now**.

To see the results of your transfer, go to **Data Workbench > Databases**.

# Import from Google Ads V2 via Workflow

You can import data from Google Ads by using td_load>: operator of workflow. If you have already created a SOURCE, you can run it; if you don't want to create a SOURCE, you can import it using a yml file.

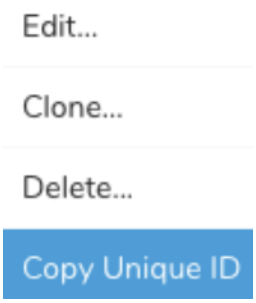
## Using a Source or YML File

### Using a Source

1. Identify your source.

2. To obtain a unique ID, open the Source list and then filter by <product>.

3. Open the menu and select "Copy Unique ID".




4. Define a workflow task using td_load> operator.

```
+load:
  td_load>: unique_id_of_your_source
  database: ${td.dest_db}
  Table: ${td.dest_table}
```

5. Run a workflow.

### Using a yml file

1. Identify your yml file. If you need to create the yml file, review Amazon S3 Import Integration Using CLI for reference.

2. Define a workflow task using td_load> operator.

```
+load:
  td_load>: config/daily_load.yml
  database: ${td.dest_db}
  Table: ${td.dest_table}
```

### 3. Run a workflow

#### Parameters Reference

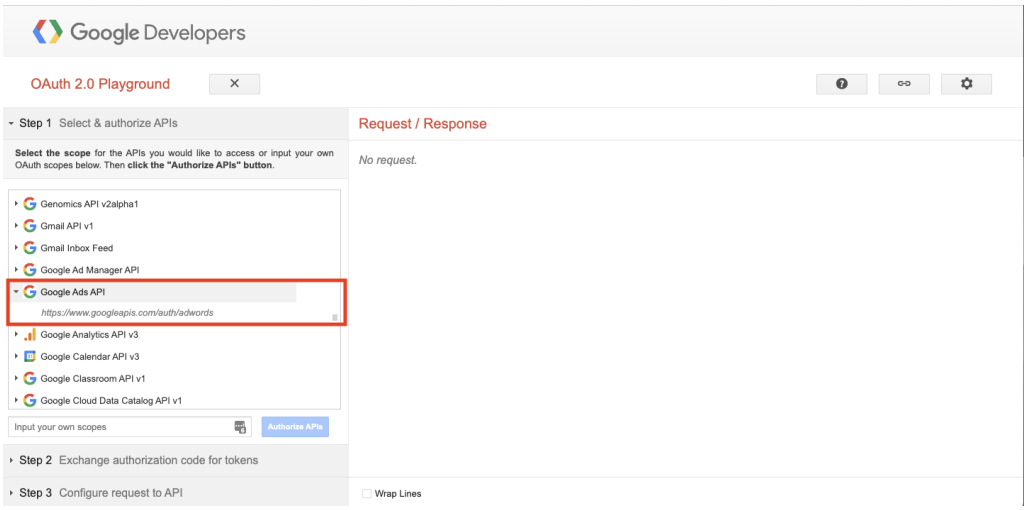
| Name | Description | Value | Default Value | Required |
|---|---|---|---|---|
| client_customer_id | Client customer ID | String | | True |
| target | Report type | String | | False |
| segments | List of additional segments | Array of String | | False |
| metrics | List of additional metrics | Array of String | | False |
| attributes | List of additional attributes | Array of String | | False |
| include_zero_impressions | Filter the records by impression | Boolean | True | False |
| include_predefined_metrics | Include all predefined metrics | Boolean | True | False |
| incremental | Run the job in incremental mode | Boolean | False | False |
| start_date | Start date, used with `date_range` is `custom_date` | Date | | False |
| end_date | End date, used with `date_range` is `custom_date` | Date | | False |
| include_negative_keywords | Filter the records by negative keywords, only used with keywords_performance_report | False | False | False |
| refresh_token | Refresh token | String | | True |
| client_id | Client ID | String | | True |
| client_secret | Client Secret | String | | True |
| developer_token | Developer Token | String | | True |
| enable_custom_query | Enable to use the custom query | Boolean | False | True |
| select_columns | List of fields to query, separated by comma | String | | False |
| from_target | Report target name | String | | False |
| other_conditions | The other condition of the query | String | | False |

### Sample Workflow Code

Visit Treasure Boxes for sample workflow code.

# Import from Google Ads V2 via CLI (Toolbelt)

Before setting up the connector, install the most current TD Toolbelt.
Please refer to this page to generate credentials on the Google side. Please make sure to choose Google Ads API in step 1 Select & authorize APIs in OAuth 2.0 playground.




## Create Seed Configuration File (seed.yml)

Unable custom query

```
in:
  type: google_ads_v2
  enable_custom_query: false
  client_id: xxx
  client_secret: xxx
  refresh_token: xxx
  client_customer_id: xx-xxxx-xxxx
  target: AD_PERFORMANCE_REPORT
  date_range: "CUSTOM_DATE"
  include_predefined_metrics: false
  developer_token: xxx
  incremental: true
  start_date: 2020-03-01
  end_date: 2020-03-02
  segments: ["segments.date"]
  metrics: ["metrics.absolute_top_impression_percentage"]
out:
  mode: append
```

Enable custom query

```
in:
  type: google_ads_v2
  enable_custom_query: true
  client_id: xxx
  client_secret: xxx
  refresh_token: xxx
  client_customer_id: xx-xxxx-xxxx
  date_range: "CUSTOM_DATE"
  developer_token: xxx
  incremental: true
  start_date: 2020-03-01
  end_date: 2020-03-02
  select_columns: ad_group_criterion.criterion_id, ad_group.id, ad_group.name, segments.date
  from_target: keyword_view
  other_conditions: "AND ad_group_criterion.type = 'KEYWORD'"
out:
  mode: append
```

### Parameters Reference

| Name | Description | Value | Default Value | Required |
|---|---|---|---|---|
| client_customer_id | Client customer ID | String | | True |
| target | Report type | String | | True |
| segments | List of additional segments | Array of String | | False |
| metrics | List of additional metrics | Array of String | | False |
| attributes | List of additional attributes | Array of String | | False |
| date_range | Date range type | String | | True |
| include_zero_impressions | Filter the records by impression | Boolean | True | False |
| include_predefined_metrics | Include all predefined metrics | Boolean | True | False |
| incremental | Run the job in incremental mode | Boolean | False | False |
| start_date | Start date, used with `date_range` is `custom_date` | Date | | False |
| end_date | End date, used with `date_range` is `custom_date` | Date | | False |
| include_negative_keywords | Filter the records by negative keywords, only used with keywords_performance_report | False | False | False |
| refresh_token | Refresh token | String | | True |
| client_id | Client ID | String | | True |
| client_secret | Client Secret | String | | True |
| developer_token | Developer Token | String | | |
| enable_custom_query | Enable to use the custom query | Boolean | False | True |
| select_columns | List of fields to query, separated by comma | String | | False |
| from_target | Report target name | String | | False |
| other_conditions | The other condition of the query | String | | False |

The data connector imports all files that match the specified prefix.

### Example

path_prefix: path/to/sample_ –> path/to/sample_201501.csv.gz, path/to/sample_201502.csv.gz, …, path/to/sample_201505.csv.gz

## Generate load.yml

Use *connector:guess*. This command automatically reads the source files and uses logic to guess the file format and its field/columns.

```
$ td connector:guess seed.yml -o load.yml
```

You can open the load.yml to review the file format definitions including file formats, encodings, column names, and types.

### Example

Disable custom query

```
in:
  type: google_ads_v2
  enable_custom_query: false
  client_id: xxx
  client_secret: xxx
  refresh_token: xxx
  client_customer_id: xx-xxxx-xxxx
  target: AD_PERFORMANCE_REPORT
  date_range: "CUSTOM_DATE"
  include_predefined_metrics: false
  developer_token: xxx
  incremental: true
  start_date: 2020-03-01
  end_date: 2020-03-02
  segments: ["segments.date"]
  metrics: ["metrics.absolute_top_impression_percentage"]
out:
  mode: append
```

Enable custom query

```
in:
  type: google_ads_v2
  enable_custom_query: true
  client_id: xxx
  client_secret: xxx
  refresh_token: xxx
  client_customer_id: xx-xxxx-xxxx
  date_range: "CUSTOM_DATE"
  developer_token: xxx
  incremental: true
  start_date: 2020-03-01
  end_date: 2020-03-02
  select_columns: ad_group_criterion.criterion_id, ad_group.id, ad_group.name, segments.date
  from_target: keyword_view
  other_conditions: "AND ad_group_criterion.type = 'KEYWORD'"
out:
  mode: append
```

To preview the data, use the *td connector:preview* command.

```
$ td connector:preview load.yml
+-------+---------+----------+---------------------+
| id    | company | customer | created_at          |
+-------+---------+----------+---------------------+
| 11200 | AA Inc. |    David | 2015-03-31 06:12:37 |
| 20313 | BB Imc. |      Tom | 2015-04-01 01:00:07 |
| 32132 | CC Inc. | Fernando | 2015-04-01 10:33:41 |
| 40133 | DD Inc. |    Cesar | 2015-04-02 05:12:32 |
| 93133 | EE Inc. |     Jake | 2015-04-02 14:11:13 |
+-------+---------+----------+---------------------+
```

The guess command requires more than 3 rows and 2 columns in the source data file because the command assesses the column definition using sample rows from the source data.

If the system detects your column name or column type unexpectedly, modify the load.yml file and preview again.

## Execute Load Job

Submit the load job.
It might take a couple of hours depending on the size of the data. Be sure to specify the Treasure Data database and table where the data should be stored.

Treasure Data also recommends specifying *--time-column* option because Treasure Data's storage is partitioned by time (see data partitioning). If this option is not provided, the data connector chooses the first *long* or *timestamp* column as the partitioning time. The type of the column specified by *--time-column* must be either of *long* and *timestamp* type.

If your data doesn't have a time column, you can add a time column by using *add_time* filter option. For more details see add_time filter plugin.

```
$ td connector:issue load.yml --database td_sample_db --table td_sample_table \
  --time-column created_at
```

The connector:issue command assumes that you have already created a *database(td_sample_db)*and a *table(td_sample_table)*. If the database or the table does not exist in TD, this command fails. Create the database and table manually or use *--auto-create-table* option with *td connector:issue* command to auto-create the database and table.

```
$ td connector:issue load.yml --database td_sample_db --table td_sample_table
 --time-column created_at --auto-create-table
```

The data connector does not sort records on the server side. To use time-based partitioning effectively, sort records in files beforehand.

If you have a field called *time*, you don't have to specify the *--time-column* option.

```
$ td connector:issue load.yml --database td_sample_db --table td_sample_table
```

### Import Modes

You can specify file import mode in the out section of the load.yml file. The out: section controls how data is imported into a Treasure Data table. For example, you may choose to append data or replace data in an existing table in Treasure Data.

| Mode | Description | Examples |
|---|---|---|
| Append | Records are appended to the target table. | in:<br>...<br>out:<br>mode: append |
| Always Replace | Replaces data in the target table.<br>Any manual schema changes made to the target table remain intact. | in:<br>...<br>out:<br>mode: replace |
| Replace on new data | Replaces data in the target table only when there is new data to import. | in:<br>...<br>out:<br>mode: replace_on_new_data |

## Scheduling Executions

You can schedule periodic data connector execution for incremental file import. Treasure Data configures our scheduler carefully to ensure high availability.

For the scheduled import, you can import all files that match the specified prefix and one of these fields by condition:

- If use_modified_time is disabled, the last path is saved for the next execution. On the second and subsequent runs, the connector only imports files that come after the last path in alphabetical order.
- Otherwise, the time that the job is executed is saved for the next execution. On the second and subsequent runs, the connector only imports files that were modified after that execution time in alphabetical order.

### Create a Schedule Using the TD Toolbelt

A new schedule can be created using the *td connector:create* command.

```
$ td connector:create daily_import "10 0 * * *" \
    td_sample_db td_sample_table load.yml
```

Treasure Data also recommends that you specify the *--time-column* option, because Treasure Data's storage is partitioned by time (see also data partitioning).

```
$ td connector:create daily_import "10 0 * * *" \
    td_sample_db td_sample_table load.yml \
    --time-column created_at
```

The `cron` parameter also accepts three special options: `@hourly`, `@daily`, and `@monthly`.

By default, the schedule is set up in the UTC timezone. You can set the schedule in a timezone using -t or --timezone option. `--timezone` option supports only extended timezone formats like 'Asia/Tokyo', 'America/Los_Angeles', etc. Timezone abbreviations like PST, CST are *not* supported and might lead to unexpected schedules.

## List All Schedules

You can see the list of currently scheduled entries by running the command *td connector:list*.

```
$ td connector:list
+--------------+--------------+----------+-------+--------------+-----------------+-------------------------
------+
| Name         | Cron         | Timezone | Delay | Database     | Table           |
Config                                |
+--------------+--------------+----------+-------+--------------+-----------------
+---------------------------------+
| daily_import | 10 0 * * *   | UTC      | 0     | td_sample_db | td_sample_table | {"in"=>{"type"=>"s3",
"access_key_id"... |
+--------------+--------------+----------+-------+--------------+-----------------
+---------------------------------+
```

## Show Schedule Settings and History

*td connector:show* shows the execution setting of a schedule entry.

| Name | Description |
|---|---|
| | |
| | |
| | |

```
% td connector:show daily_import
Name     : daily_import
Cron     : 10 0 * * *
Timezone : UTC
Delay    : 0
Database : td_sample_db
Table    : td_sample_table
Config
---
in:
  type: google_ads_v2
  enable_custom_query: false
  client_id: xxx
  client_secret: xxx
  refresh_token: xxx
  client_customer_id: xx-xxxx-xxxx
  target: AD_PERFORMANCE_REPORT
  date_range: "CUSTOM_DATE"
  include_predefined_metrics: false
  developer_token: xxx
  incremental: true
  start_date: 2020-03-01
  end_date: 2020-03-02
  segments: ["segments.date"]
  metrics: ["metrics.absolute_top_impression_percentage"]      ...
```

*td connector:history* shows the execution history of a scheduled entry. To investigate the results of each individual run, use *td job <jobid>*.

```
% td connector:history daily_import
+--------+---------+---------+--------------+-----------------+----------+---------------------------+----------
+
| JobID  | Status  | Records | Database     | Table           | Priority | Started                   | Duration
|
+--------+---------+---------+--------------+-----------------+----------+---------------------------+----------
+
| 578066 | success | 10000   | td_sample_db | td_sample_table | 0        | 2015-04-18 00:10:05 +0000 | 160
|
| 577968 | success | 10000   | td_sample_db | td_sample_table | 0        | 2015-04-17 00:10:07 +0000 | 161
|
| 577914 | success | 10000   | td_sample_db | td_sample_table | 0        | 2015-04-16 00:10:03 +0000 | 152
|
| 577872 | success | 10000   | td_sample_db | td_sample_table | 0        | 2015-04-15 00:10:04 +0000 | 163
|
| 577810 | success | 10000   | td_sample_db | td_sample_table | 0        | 2015-04-14 00:10:04 +0000 | 164
|
| 577766 | success | 10000   | td_sample_db | td_sample_table | 0        | 2015-04-13 00:10:04 +0000 | 155
|
| 577710 | success | 10000   | td_sample_db | td_sample_table | 0        | 2015-04-12 00:10:05 +0000 | 156
|
| 577610 | success | 10000   | td_sample_db | td_sample_table | 0        | 2015-04-11 00:10:04 +0000 | 157
|
+--------+---------+---------+--------------+-----------------+----------+---------------------------+----------
+
8 rows in set
```

## Delete Schedule

*td connector:delete* removes the schedule.

```
$ td connector:delete daily_import
```

## How to convert Google Ads Query Language to custom query in the connector

Query language reference: https://developers.google.com/google-ads/api/docs/query/overview?hl=en

Example:

```
SELECT
  campaign.id,
  campaign.name,
  campaign.status,
  metrics.impressions,
  segments.date,
FROM campaign
WHERE segments.date during LAST_30_DAYS
  AND campaign.status = 'PAUSED'
  AND metrics.impressions > 1000
ORDER BY campaign.id
```

```
in:
  type: google_ads_v2
  enable_custom_query: true
  client_id: xxx
  client_secret: xxx
  refresh_token: xxx
  client_customer_id: xx-xxxx-xxxx
  date_range: "LAST_30_DAYS"
  developer_token: xxx
  select_columns: campaign.id, campaign.name, campaign.status, metrics.impressions, segments.date
  from_target: campaign
  other_conditions: "AND campaign.status = 'PAUSED' AND metrics.impressions > 1000"
```

Note: The query should have time and it presents the where condition, `other_conditions` should start with `AND` as in the example.