

Bulk Import from Google Sheets

This article explains how to import data from Google Sheets to Treasure Data using `embulk-input-google_spreadsheets` input plugin.

Continue to the following topics:

- [Prerequisites](#)
- [Install `embulk-input-google_spreadsheets` Plugin](#)
- [Obtain Required Google API Credentials](#)
 - [Add Google OAuth 2.0 Playground Redirect URI](#)
- [Create Embulk Configuration File](#)
- [Execute Load Job](#)



This content is obsolete and will no longer be updated. Treasure Data recommends using our native integration for [Google Sheets](#).

Prerequisites

- Basic knowledge of Treasure Data
- Basic knowledge of [Embulk](#).
- Embulk is a Java application. Make sure that Java is installed.
- Follow the instructions in [Installing Bulk Data Import](#).
- [Embulk and `embulk-output-td` plugin](#) installed on your machine.

Install `embulk-input-google_spreadsheets` Plugin

To install `embulk-input-google_spreadsheets` plugin, run the following command:

```
$ embulk gem install embulk-input-google_spreadsheets
```

Obtain Required Google API Credentials

For Embulk to connect to the Google Sheets API there are different authentication options available. This example will use the `authorized_user` method which requires a JSON key file. You can get the key file from the Google API console. The key file contains the credentials to allow connection to Google Sheets API. It has three fields required for the Embulk plugin to run.

```
* client_id
* client_secret
* refresh_token
```

You will need to download the key file from an existing Google API project. If you have not created a project, you can do so using this wizard provided by Google that will allow you to create a project and turn on the Google Sheets API [Google Sheets API Wizard](#).

Go to Credentials > Create Credentials > OAuth ClientID > Web Application. Enter a name, then 'Create'. The next screen will show you the client ID and client secret. Copy them as you will need them for the JSON key file. Completing these steps will provide you with the `client_id` and `client_secret`.

Add Google OAuth 2.0 Playground Redirect URI

You must obtain a refresh token from the Google OAuth 2.0 Playground. You must add the OAuth 2.0 Playground Redirect URI (<https://developers.google.com/oauthplayground>) to the Authorized Redirect URI's for the credentials you created.

1. Select the pencil icon to edit the credentials you are going to use for the Google OAuth 2.0 Playground.

Credentials

Credentials OAuth consent screen Domain verification

Create credentials

Create credentials to access your enabled APIs. For more information, see the [authentication documentation](#).

OAuth 2.0 client IDs

<input type="checkbox"/>	Name	Creation date	Type	Client ID			
<input type="checkbox"/>	SheetsFriday	Mar 26, 2019	Web application	142714532728-ccdb8f9c305e30a9926b4c32a7f1...			  



1. In the Authorized Redirect URI field, enter the url for the Auth Playground: <https://developers.google.com/oauthplayground>. Press enter and SELECT save.

Client ID	142714532728-ccdb8f9c305e30a9926b4c32a7f1..._apps.googleusercontent.com
Client secret	22222222222222222222222222222222
Creation date	Mar 26, 2019 10:00:00 AM

Name

SheetsFriday

Restrictions

Enter JavaScript origins, redirect URIs, or both [Learn More](#)

Origins and redirect domains must be added to the list of Authorized Domains in the [OAuth consent settings](#).



Authorized JavaScript origins

For use with requests from a browser. This is the origin URI of the client application. It can't contain a wildcard (https://*.example.com) or a path (<https://example.com/subdir>). If you're using a nonstandard port, you must include it in the origin URI.

Type in the domain and press Enter to add it

Authorized redirect URIs

For use with requests from a web server. This is the path in your application that users are redirected to after they have authenticated with Google. The path will be appended with the authorization code for access. Must have a protocol. Cannot contain URL fragments or relative paths. Cannot be a public IP address.

Type in the domain and press Enter to add it

The remaining credential you need is the refresh token. One way to get the refresh token is to use the Google OAuth 2.0 Playground available at [Google OAuth 2.0 Playground](#).

In the top right select the Gear Icon and make sure to select the Use your own OAuth credentials. In the OAuth Client ID and the OAuth Client secret fields, insert the credentials you got from the Google API console.

From the API list on the left, select the Google Sheets API v4. Then proceed to the Step 2. - Exchange authorization code for tokens. Press the Exchange authorization code for tokens button which will fill the Refresh Token and Access Token fields. The refresh token field is the value you should note down for the JSON key file Embulk will require. With all components of the JSON key file collected, you are ready to begin the Embulk configuration. The format of the JSON key file is:

```
{
  "client_id": "xxxxxxxxxx",
  "client_secret": "xxxxxxxxxx",
  "refresh_token": "xxxxxxxxxx"
}
```

Create Embulk Configuration File

Using your favorite text editor, create the Embulk config file defining input (Google Sheets) and output (TD) parameters.

In the example below, the required JSON key file is used in-line. For further details about additional parameters and to view other examples, refer to [Embulk Input Google Sheets](#).

The fields that are required for the plugin to run are `spreadsheets_url` and `worksheet_title`. Edit the relevant details for the sheet you are trying to import. The same applies for the output section which outputs to Treasure Data.

The example assumes that you have already created a relevant table in Treasure Data that matches the details used in the config file. For example database would be the database created for the data, table would be the table within that database to receive the data.

```
in:
  type: google_spreadsheets
  auth_method: authorized_user
  json_keyfile:
    content: |
      {
        "client_id": "xxxxxxxxxx",
        "client_secret": "xxxxxxxxxx",
        "refresh_token": "xxxxxxxxxx"
      }
  spreadsheets_url: https://docs.google.com/spreadsheets/d/1xP_ZBA-Or4sJ63Zj99-JreuAFGn7Y2gcUKd1vtwSqwM
/edit#gid=0
  worksheet_title: title
  start_row: starting row
  default_timezone: timezone to be used
  null_string: '\N'
  default_typecast: strict
  columns:
    - {name: color, type: string}
    - {name: size, type: string}
    - {name: type, type: string}
out:
  type: td
  apikey: xxxxxxxxxxxx
  endpoint: api.treasuredata.com
  database: dbname
  table: tblname
  time_column: datecolumn
  mode: replace
  #by default mode: append is used, if not defined. Imported records are appended to the target table with
  this mode.
  #mode: replace, replaces existing target table
  default_timestamp_format: '%d/%m/%Y'
```

Add the "auto_create_table: true" parameter to the load.yml, so that tables that do not exist are automatically created.

This is an example of the auto_create_table parameter in a .yml file.

```
out:
  type: td
  apikey: <your apikey>
  endpoint: api.treasuredata.com
  database: dbname
  table: tblname
  time_column: created_at
  auto_create_table: true
  mode: append
```

You must create the database and table in TD, prior to executing the load job. Alternatively, if you: 1) must add a database or 2) do not add the `auto_create_table` parameter in a `.yaml` file and must add a table, run the following TD commands:

```
$ td database:create dbname
$ td table:create dbname tblname
```

You can also create the database and table using [TD Console](#).

Execute Load Job

Issue the import job by running the following command:

```
$ embulk run load.yaml
```

The run time of the import job varies depending on the size of the data you are importing. After the job is done, the data should reflect in the TD database and table used in the Embulk configuration file.