KARTE Import Integration

Learn more about KARTE Export Integration.

You can use the KARTE Data Connector to import the contents of *.tsv and* .csv files stored in your Google Cloud Storage (GCS) bucket into Treasure Data.

Prerequisites

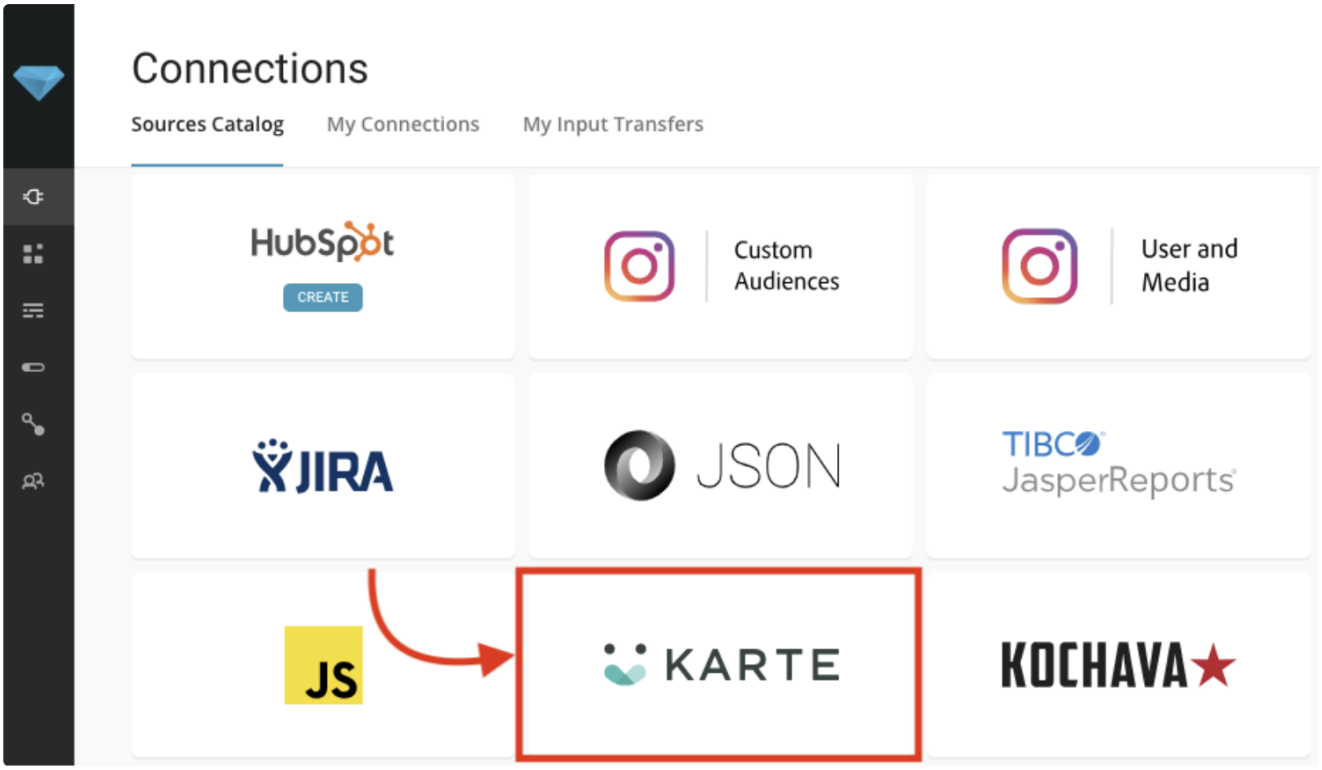
- Basic knowledge of Treasure Data
- An existing Google Service Account
    - You also need to generate and obtain a JSON key file from Google Developers Console. See the Generating a service account credential section of Google Cloud Platform's documentation for additional information.

Use the TD Console to Create Your Connection

You can use the TD Console to configure your connection.

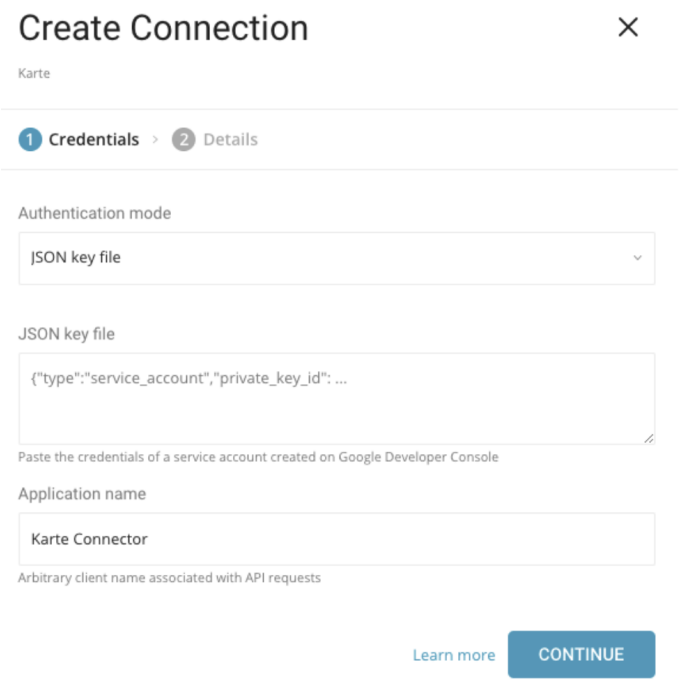
Create a New Connection

Go to **Integrations Hub** > **Catalog** and search. Select KARTE.



Create a New KARTE Connector

The following dialog opens.



Provide the required credentials. In the TD Console, you can use only a JSON key file to authenticate:

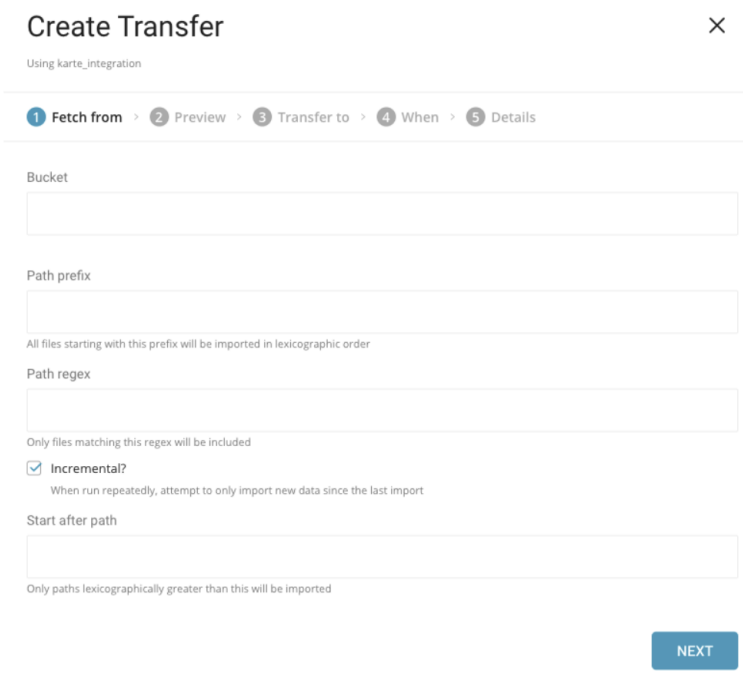
- **JSON key file:** Copy and paste the contents of the JSON key file generated from the Google Developers Console into this field.
- **Application Name:** *KARTE* is the default value. KARTE is the Treasure Data chosen name for this connector.

Select **Continue**.

Specify a name for your data connector and select **Done**.

Import KARTE Data to Treasure Data

The Connections page displays. Select **New Transfer**.



The following sections provide instructions to help you fetch, preview, transfer, schedule, and review details.

Fetch from

You need to register the information that you would like to ingest:

- **Bucket**. Google Cloud Storage bucket name (Example: *your_bucket_name*).
- **Path Prefix**. The prefix for target keys. (Example: *logs/data_*).
- **Path Regex**. Specify the regexp to match file paths. If a file path doesn't match with this pattern, the file is skipped. (Example: In the case of *.csv$ #*, a file is skipped if its path doesn't match with this pattern.).
- **Start after path**. Inserts the last_path parameter so that the first execution skips files before the path. (Example: logs /data_20170101.csv).
- **Incremental**. Enables incremental loading. If incremental loading is enabled, the config diff for the next execution will include the last_path parameter and the next execution skips files before the path. Otherwise, last_path will not be included.

Example

Amazon CloudFront is a web service that speeds up the distribution of your static and dynamic web content. You can configure CloudFront to create log files that contain detailed information about every user request that CloudFront receives. If you enable logging, you can save CloudFront log files, shown as follows:

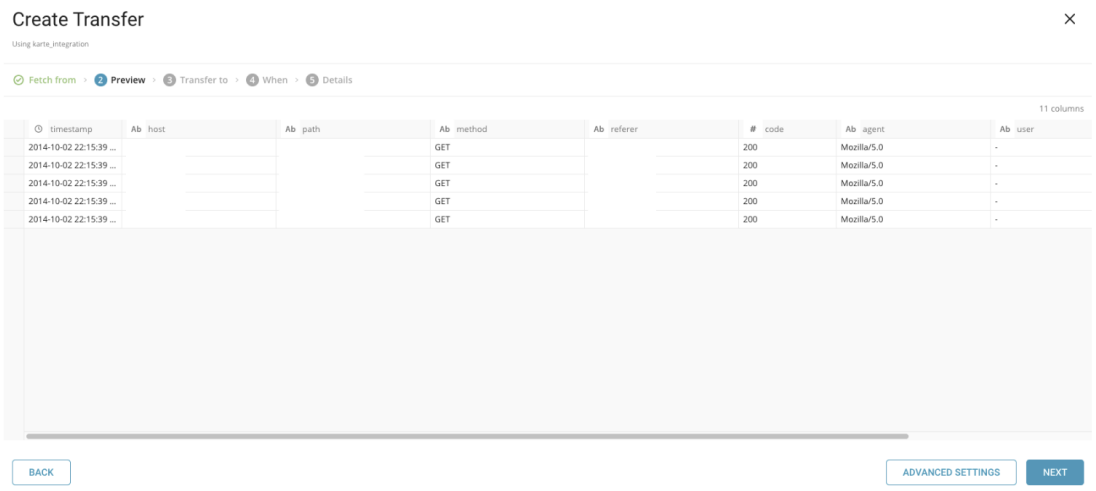
```
[your_bucket] - [logging] - [E231A697YXWD39.2017-04-23-15.a103fd5a.gz]
[your_bucket] - [logging] - [E231A697YXWD39.2017-04-23-15.b2aede4a.gz]
[your_bucket] - [logging] - [E231A697YXWD39.2017-04-23-16.594fa8e6.gz]
[your_bucket] - [logging] - [E231A697YXWD39.2017-04-23-16.d12f42f9.gz]
```

In this case, **Fetch from** setting should be as follows:

- **Bucket**: your_bucket
- **Path Prefix**: logging/
- **Path Regex**: *.gz$* (Not Required)
- **Start after path**: logging/E231A697YXWD39.2017-04-23-15.b2aede4a.gz (Assuming that you want to import the logfiles from 2017-04-23-16.)
- **Incremental**: true (if you want to schedule this job.)

Preview

You'll see a preview of your data. For more information on how data preview works, review About Data Preview.
To make changes, such as set specified column name, select **Advanced Settings** otherwise, select **Next**.



Advanced Settings

Advanced Settings allow you to edit properties. The following properties are available:

- **Default timezone**. Changes the time zone of timestamp columns if the value itself doesn't include time zone.
- **Columns**
    - **Name**. Changes the name of the column. The column name supports lowercase alphabets, numbers, and "_" only.
    - **Type**. Type parses a value as a specified type and then stores the type after converting to TreasureData schema.
        - **boolean**
        - **long**
        - **timestamp**: Imported as string type at TreasureData (Ex. 2017-04-01 00:00:00.000)
        - **double**
        - **string**
        - **json**
- **Total file count limit**. The maximum number of files to read. (optional)

Transfer to

Choose an existing or create a new database and table for the import.

- **Mode**: Append or Replace. Select whether to append records to an existing table or replace your existing table.
- **Partition Key Seed**: Choose the long or timestamp column as the partitioning time. As the default time column, it uses the **upload_time** with the **add_time** filter.

When

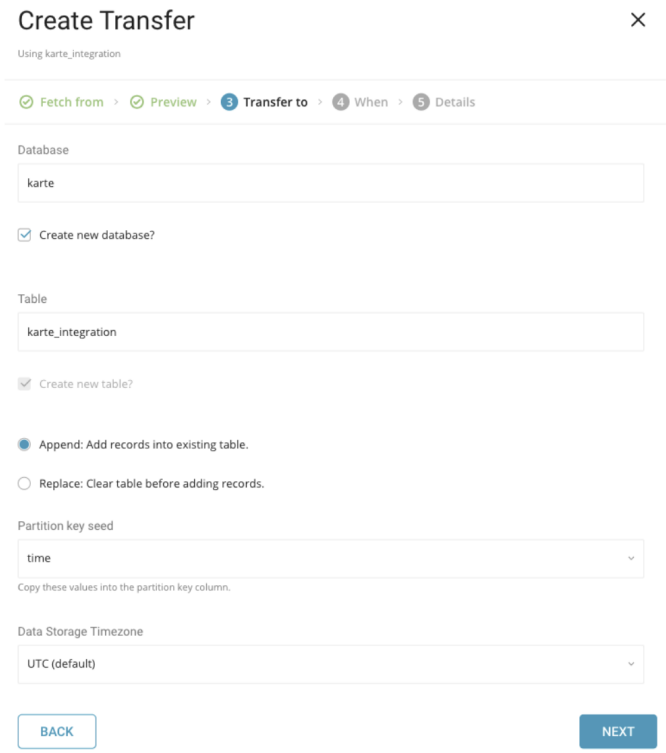
You can specify a one-time transfer, or you can schedule an automated recurring transfer.

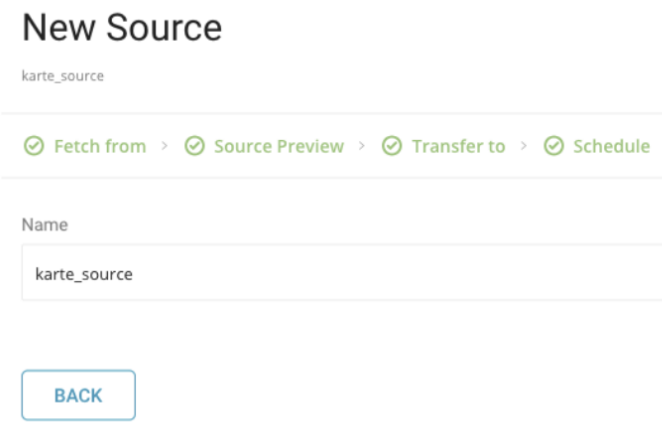
Use the following parameters to set your schedule:

- **Once now**. Run the job one time.
- **Repeat**
    - **Schedule**. You can schedule with three options: *@hourly*, *@daily* and *@monthly* and custom *cron*.
    - **Delay Transfer**. Add a delay to the specified run time.
- **Timezone**. supports extended time zone formats, for example, Asia/Tokyo.

Details

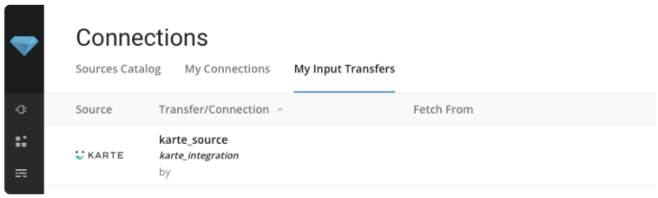
Create a name for your source and then select **DONE** to save your KARTE connector as a new source.

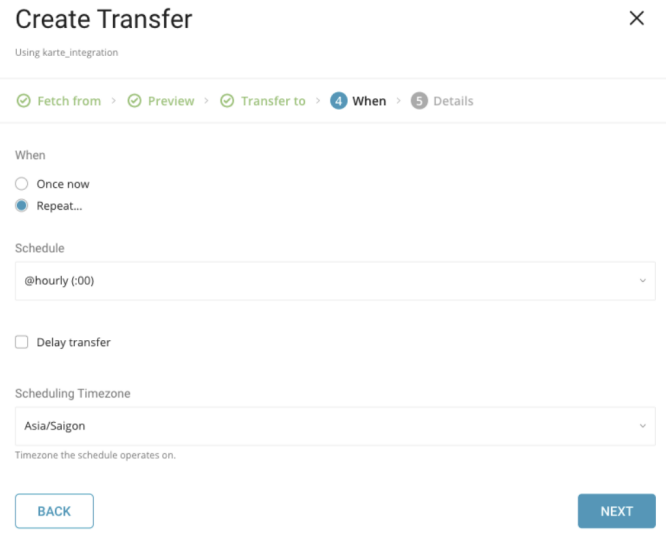




My Input Transfers

Edit your existing jobs in My Input Transfers. To view the details of previous transfers for this data connector, select the jobs icon.





Using the CLI to Configure the Connector

Before setting up the connector, install the Treasure Data Toolbelt.

Create Seed Config File (seed.yml)

Prepare *seed.yml* with the following example and your JSON key file. You must also specify the bucket name and target file name (or prefix for multiple files).

```
in:
  type: karte
  bucket: sample_bucket
  path_prefix: path/to/sample_file  # path the the *.csv or *.tsv file on your GCS bucket
  auth_method: json_key
  json_keyfile:
    content: |
      {
        "private_key_id": "1234567890",
        "private_key": "-----BEGIN PRIVATE KEY-----\nABCDEF",
        "client_id": "...",
        "client_email": "...",
        "type": "service_account"
      }
out:
  mode: append
```

The Data Connector for KARTE imports all files that match a specified prefix. For example, path_prefix: `path/to/sample_` –> `path/to/sample_201501.csv.gz`, `path/to/sample_201502.csv.gz`, ..., `path/to/sample_201505.csv.gz`.

For more details on available *out* modes, see the Appendix below.

Guess Fields (Generate load.yml)

Use *connector:guess*. This command automatically reads the target file, and intelligently guesses the file format.

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

Open up *load.yml* to see the guessed file format definitions including file formats, encodings, column names, and types.

```
in:
  type: karte
  bucket: sample_bucket
  path_prefix: path/to/sample_file
  auth_method: json_key
  json_keyfile:
    content: |
      {
        "private_key_id": "1234567890",
        "private_key": "-----BEGIN PRIVATE KEY-----\nABCDEF",
        "client_id": "...",
        "client_email": "...",
        "type": "service_account"
      }
  decoders:
  - {type: gzip}
  parser:
    charset: UTF-8
    newline: CRLF
    type: csv
    delimiter: ','
    quote: '"'
    escape: ''
    skip_header_lines: 1
    columns:
    - name: id
      type: long
    - name: company
      type: string
    - name: customer
      type: string
    - name: created_at
      type: timestamp
      format: '%Y-%m-%d %H:%M:%S'
out:
  mode: append
```

Preview how the system parses the file by using the *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 it guesses column definition using sample rows from source data.

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

> The Data Connector supports parsing of "boolean", "long", "double", "string", and "timestamp" types.

The preview command downloads one file from the specified bucket and displays the results from that file. This might cause a difference in results from the preview and issue commands.

Execute Load Job

Finally, submit the load job. It might take a couple of hours depending on the size of the data. Users need to specify the database and table where their data is stored.

It's also recommended to specify *--time-column* option, since Treasure Data's storage is partitioned by time (see also data partitioning). If you don't specify the option, the Data Connector selects the first *long* or *timestamp* column as the partitioning time. The type of the column specified by *--time-column* must be either of *long* or *timestamp* type.

If your data doesn't have a time column you can add it using *add_time* filter option. Find more details at add_time filter plugin.

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

The previs example command assumes you have already created *database(td_sample_db)* and *table(td_sample_table)*. If the database or the table do not exist in Treasure Data, this command cannot be successful. Instead, create the database and table manually or use *--auto-create-table* option with the *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 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
```

Scheduling Run Times

You can schedule periodic Data Connector to run for incremental KARTE file imports using our high availability scheduler. By using this feature, you no longer need a *cron* daemon on your local data center.

For the scheduled import, the Data Connector for KARTE imports all files that match with the specified prefix (e.g. path_prefix: `path/to/sample_` –> `path/to/sample_201501.csv.gz`, `path/to/sample_201502.csv.gz`, ..., `path/to/sample_201505.csv.gz`) at first and remembers the last path (`path/to/sample_201505.csv.gz`) for the next execution.

On the second and subsequent runs, it only imports files that come after the last path in alphabetical (lexicographic) order. For example, `path/to/sample_201506.csv.gz`, ...

Create the Schedule

You can create a new schedule with the *td connector:create* command. The following are required:

- The schedule name.
- The cron-style schedule.
- The database and table where their data will be stored.
- The Data Connector configuration file.

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

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

```
$ 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, a schedule is set up in the UTC timezone. You can set the schedule in a timezone using -t or --timezone option. `--timezone` option only supports 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 the Schedules

You can see the list of 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"=>"karte", ... |
+--------------+------------+----------+-------+--------------+-----------------+----------------------------+
```

Show the Settings and Schedule History

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

```
% 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: karte
  bucket: sample_bucket
  path_prefix: path/to/sample_
  auth_method: json_key
  json_keyfile:
    content: |
      {
        "private_key_id": "1234567890",
        "private_key": "-----BEGIN PRIVATE KEY-----\nABCDEF",
        "client_email": "...",
        "type": "service_account"
      }
  decoders:
  - type: gzip
  parser:
    charset: UTF-8
    ...
```

td connector:history shows the execution history of a schedule 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 the Schedule

td connector:delete will remove the schedule.

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

Appendix

Modes for Out Plugin

You can specify file import mode in *out* section of seed.yml.

append (default)

This is the default mode and records are appended to the target table.

```
in:
  ...
out:
  mode: append
```

replace (In td 0.11.10 and later)

This mode replaces data in the target table.Any manual schema changes made to the target table will remain intact with this mode.

```
in:
  ...
out:
  mode: replace
```