Microsoft Azure Blob Storage Import Integration

Learn more about Microsoft Azure Blob Storage Export Integration.

Open the Data Connector for Microsoft Azure Blob Storage enables the import of the contents of *.tsv and* .csv files stored in your Azure Blob Storage container.

Limitations

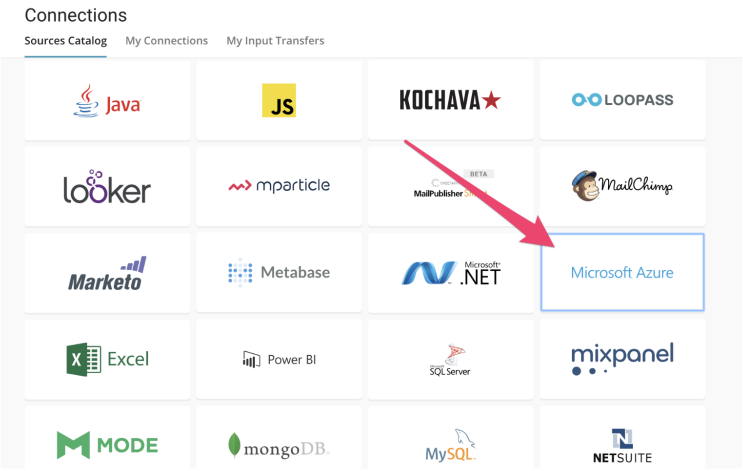
Connector UI limitations. Editing with the Connector UI has many limitations. We suggest using CLI for your edits.

Prerequisites

- Basic knowledge of Treasure Data
- A Microsoft Azure Platform account

Configure the Connection

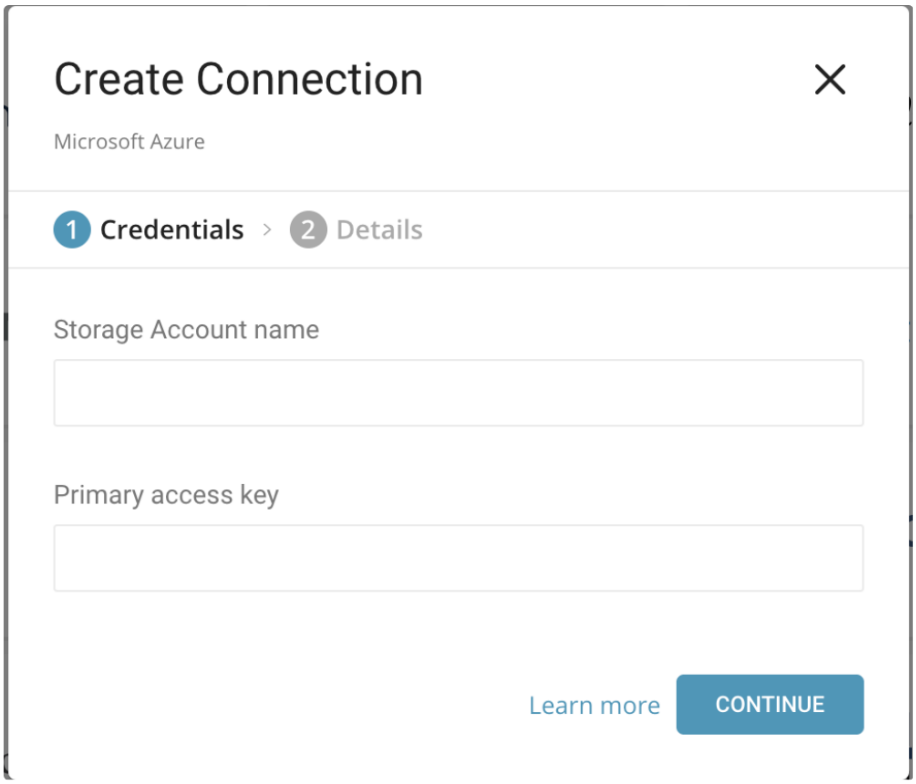
You can submit a DataConnector for Microsoft Azure Blob Storage from the Connector UI.



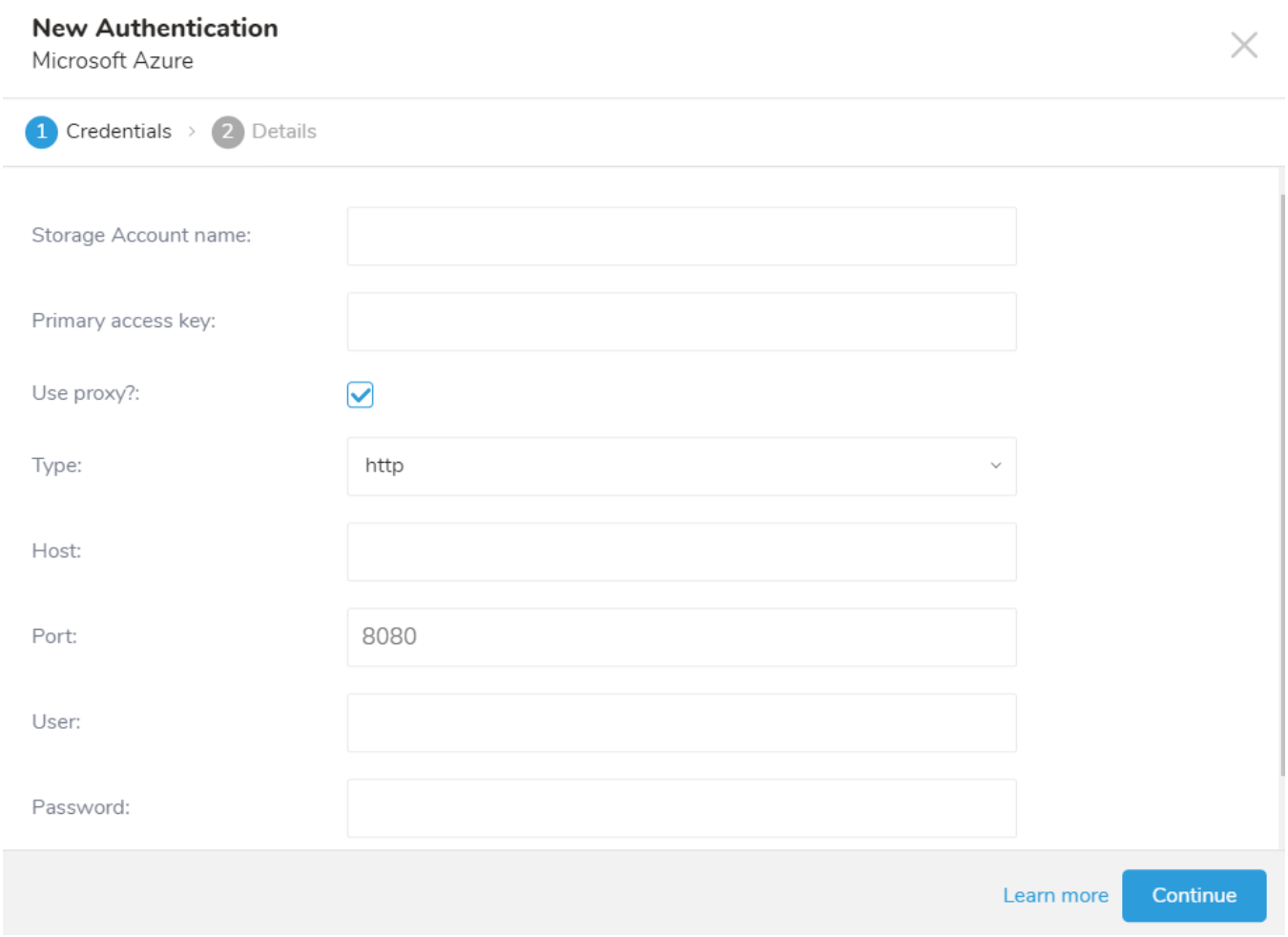
Create a new Microsoft Azure Blob Storage connector

First, you must register the connector by setting the following parameters:

- **Storage Account name**:The name of your Microsoft Azure Blob Storage account.
- **Primary access key:** The access key used to access your Microsoft Azure Blob Storage account.

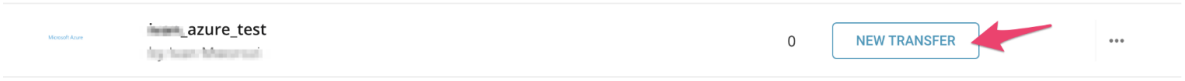


With the proxy setting enabled



Transfer data from Microsoft Azure Storage

Next, create “New Transfer” on the My Connections page. You can prepare an ad hoc DataConnector job or an schedule DataConnector job. Complete the following steps.



Fetch from

Register the information that you want to ingest.

- **Container**: Azure cloud storage container name (Ex. *your_cont*)
- **Path Prefix**: prefix of target keys. (Ex. *logs/data_*)
- **Path Regex**: regexp to match file paths. If a file path doesn’t match with this pattern, the file is skipped. (Ex. *.csv$ #* in this case, a file is skipped if its path doesn’t match with this pattern)

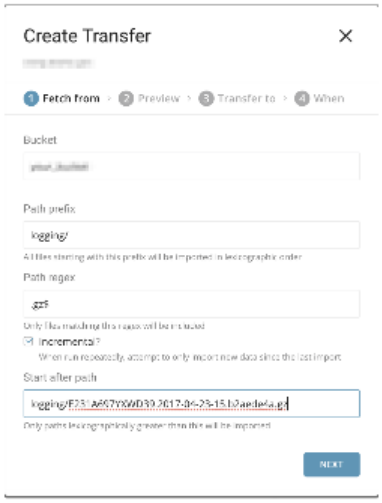
Example: CloudFront

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]
```

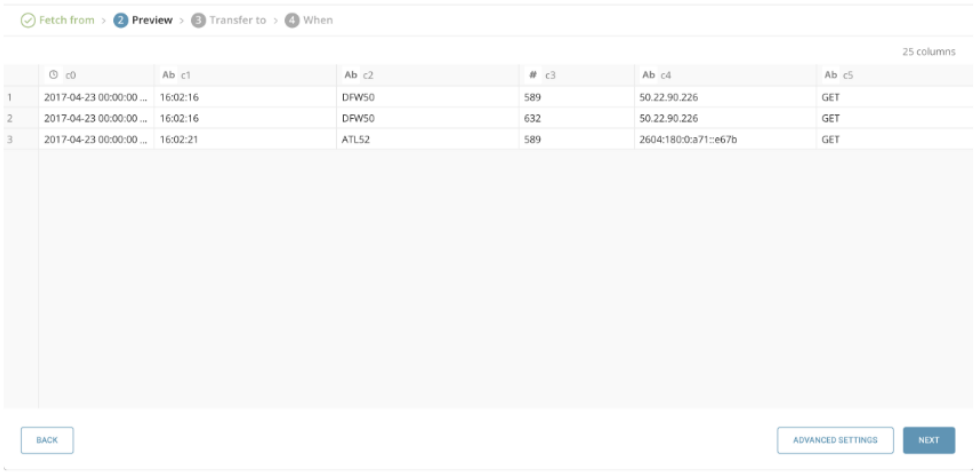
In this case, “Fetch from” setting should be as shown:

- **Container**: your_container
- **Path Prefix**: logging/
- **Path Regex**: *.gz$* (Not Required)



Preview

You can see a preview of data you configured. If you are unable to see the preview or have any issues viewing the preview, contact support.



The preview command will download one file from the specified bucket and display the results from that file. This may cause a difference in results from the preview and issue commands.

If you want to set a specified column name, select **Advanced Settings**.

Advanced Settings

Advanced Settings allows you to edit guessed properties. Edit the following section, if you need to.

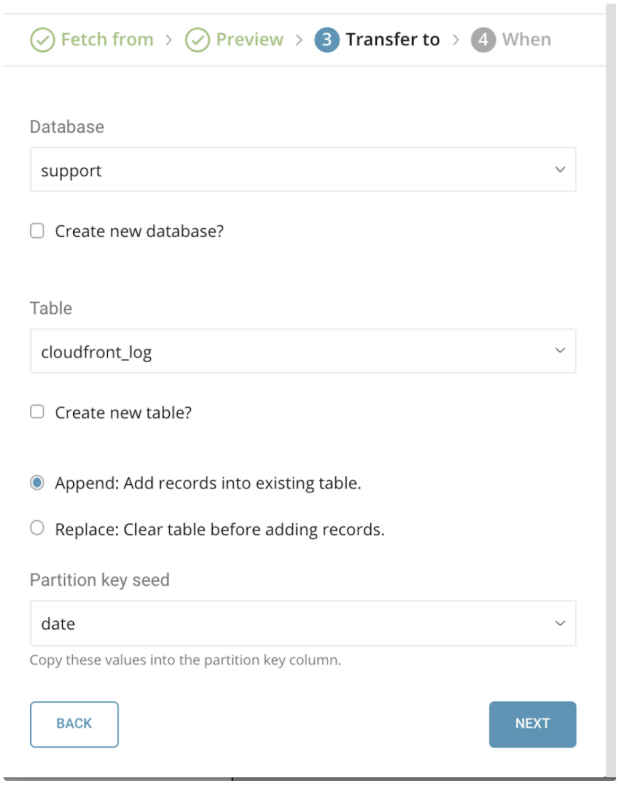
- **Default timezone**: Changes Time zone of timestamp columns if the value itself doesn’t include time zone.
- **Columns**:

- **Name**: Changes the name of the column. Supported characters for column names are lowercase alphabets, numbers, and "_" (underscore) only.
- **Type**: Parses a value as a specified type and stores the type as part of the Treasure Data schema.
    - **boolean**
    - **long**
    - **timestamp**: imported as String type at Treasure Data (Ex. 2017-04-01 00:00:00.000)
    - **double**
    - **string**
    - **json**
- **Total file count limit**: maximum number of files to read. (optional)

Transfer to

In this phase, select your target database and table that you want to import to. You can create a new database or table using the `Create new database` or `Create new table` checkboxes.

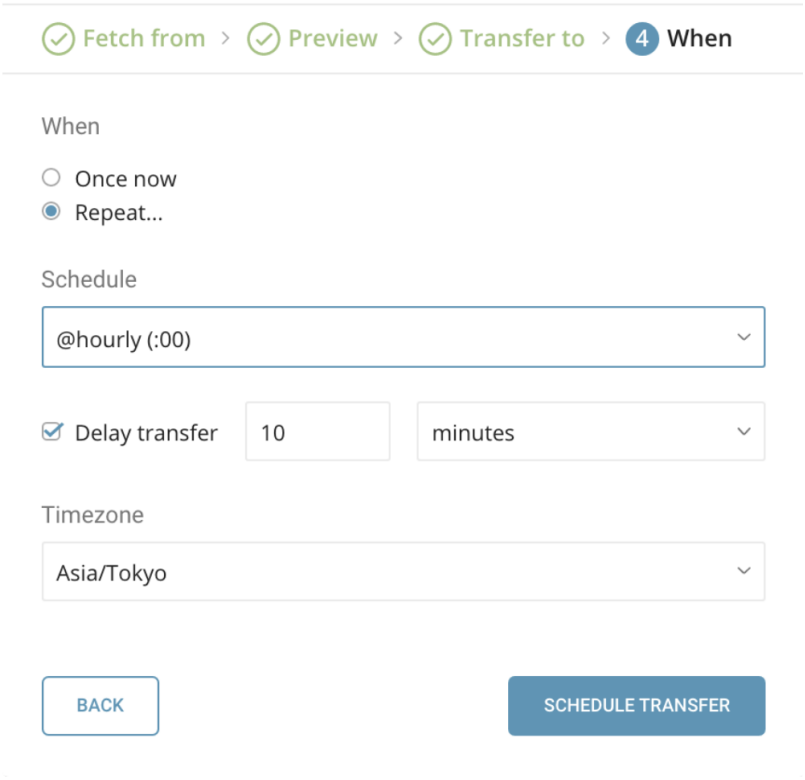
- **Mode**: Append – Allows you to add records into the existing table.
- **Mode**: Replace – Replace the existing data in the table with the data being imported.
- **Partition key Seed**: Choose the long or timestamp column that you would like to use as the partitioning time column. If you do not specify a time column, the upload time of the transfer is used in conjunction with the addition of an add_time filter.



When

In this phase, you can set an ad hoc or schedule configuration for your job.

- When
    - **Once now**: Run the transfer only once.
    - **Repeat…**
        - **Schedule**: accepts these three options: `@hourly`, `@daily`and `@monthly`and custom `cron`.
        - **Delay Transfer**: add a delay of execution time.
    - **Data Storage Timezone**: Timezone the data is stored in; data will also be displayed in this timezone. Supports extended timezone formats like 'Asia/Tokyo'.



After selecting the frequency, select **Start Transfer** to begin the transfer. If there are no errors, the transfer into Treasure Data will complete and the data will be available. Jobs are kicked off when a transfer runs. You can use the Jobs or the My Input Transfers section to monitor the progress of your data transfer.

Troubleshoot Data Import

Review the job log. Warning and errors provide information about the success of your import. For example, you can identify the source file names associated with import errors.

Use the CLI to Configure the Connector

You can also use the Microsoft Azure Blob Storage data connector from the command line interface. The following instructions show you how to import data using the CLI.

Install ‘td’ command v0.11.9 or later

Install the newest Treasure Data Toolbelt.

```
$ td --version
0.11.10
```

Create Seed Config File (seed.yml)

First, prepare *seed.yml* as shown in the following example, with your account information (Check about Azure storage accounts). You must also specify container name and target file name (or prefix for multiple files).

```
in:
  type: azure_blob_storage
  account_name: myaccount
  account_key: myaccount_key
  container: my-container
  path_prefix: logs/csv-
out:
  mode: append
```

The Data Connector for Microsoft Azure Blob Storage imports all files that match a 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`)

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

4.3. Guess Fields (Generate load.yml)

Second, use *connector:guess*. This command automatically reads the source file, and assesses (uses logic to guess) the file format.

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

If you open up *load.yml*, you'll see the guessed file format definitions including file formats, encodings, column names, and types.

```
in:
  type: azure_blob_storage
  account_name: myaccount
  account_key: myaccount_key
  container: my-container
  path_prefix: logs/csv-
  decoders:
  - {type: gzip}
  parser:
    charset: UTF-8
    newline: CRLF
    type: csv
    delimiter: ','
    quote: '"'
    header_line: true
    columns:
    - {name: id, type: long}
    - {name: account, type: long}
    - {name: time, type: timestamp, format: '%Y-%m-%d %H:%M:%S'}
    - {name: purchase, type: timestamp, format: '%Y%m%d'}
    - {name: comment, type: string}
out:
  mode: append
```

Then, you can preview how the system will parse 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 needs over 3 rows and 2 columns in 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.

Currently, the data connector supports parsing of “boolean”, “long”, “double”, “string”, and “timestamp” types.

You also must create a local database and table prior to executing the data load job. Follow these steps:

```
$ td database:create td_sample_db
$ td table:create td_sample_db td_sample_table
```

4.4. Execute Load Job

Finally, submit the load job. It may take a couple of hours depending on the size of the data. Specify the Treasure Data database and table where the data should be stored.

It’s also recommended to specify *--time-column* option, because Treasure Data’s storage is partitioned by time (see data partitioning). If the 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 do not exist in TD, the connector:issue command fails, so 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
```

At present, the data connector does not sort records on 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
```

Scheduled execution

You can schedule periodic Data Connector execution for incremental Microsoft Azure Blob Storage file imports. We configure our scheduler carefully to ensure high availability. By using this feature, you no longer need a *cron* daemon on your local data center.

For the scheduled import, the Data Connector for Microsoft Azure Blob Storage 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, the connector imports only files that comes after the last path in alphabetical (lexicographic) order. (`path/to/sample_201506.csv.gz`, ...).

Create the schedule

A new schedule can be created using the *td connector:create* command. The following are required: the name of the schedule, the cron-style schedule, the database and table where their data will be stored, and the Data Connector configuration file.

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

It's also recommended to specify the *--time-column* option, because Treasure Data's storage is partitioned by time (see 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, schedule is setup in UTC timezone. You can set the schedule in a timezone using -t or --timezone option. Note that `--timezone` option supports only extended timezone formats like 'Asia/Tokyo', 'America/Los_Angeles' etc. Timezone abbreviations like PST, CST are *not* supported and may lead to unexpected schedules.

List the 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"=>"
azure_blob_storage", ... |
+--------------+------------+----------+-------+--------------+-----------------
+-----------------------------------------+
```

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
```

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 removes the schedule.

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

Appendix

A) 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. Note that any manual schema changes made to the target table remain intact with this mode.

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

B) Proxy Setting

```
in:
  type: azure_blob_storage
  account_name: myaccount
  account_key: myaccount_key
  container: my-container
  path_prefix: logs/csv-
  proxy:
        type: http
        host: 201.202.203.10
        port: 8080
        user: test
        password: test
```