Mixpanel Import Integration

The Data Connector for Mixpanel allows you to back up the event data in Mixpanel on Treasure Data. Use cases include:

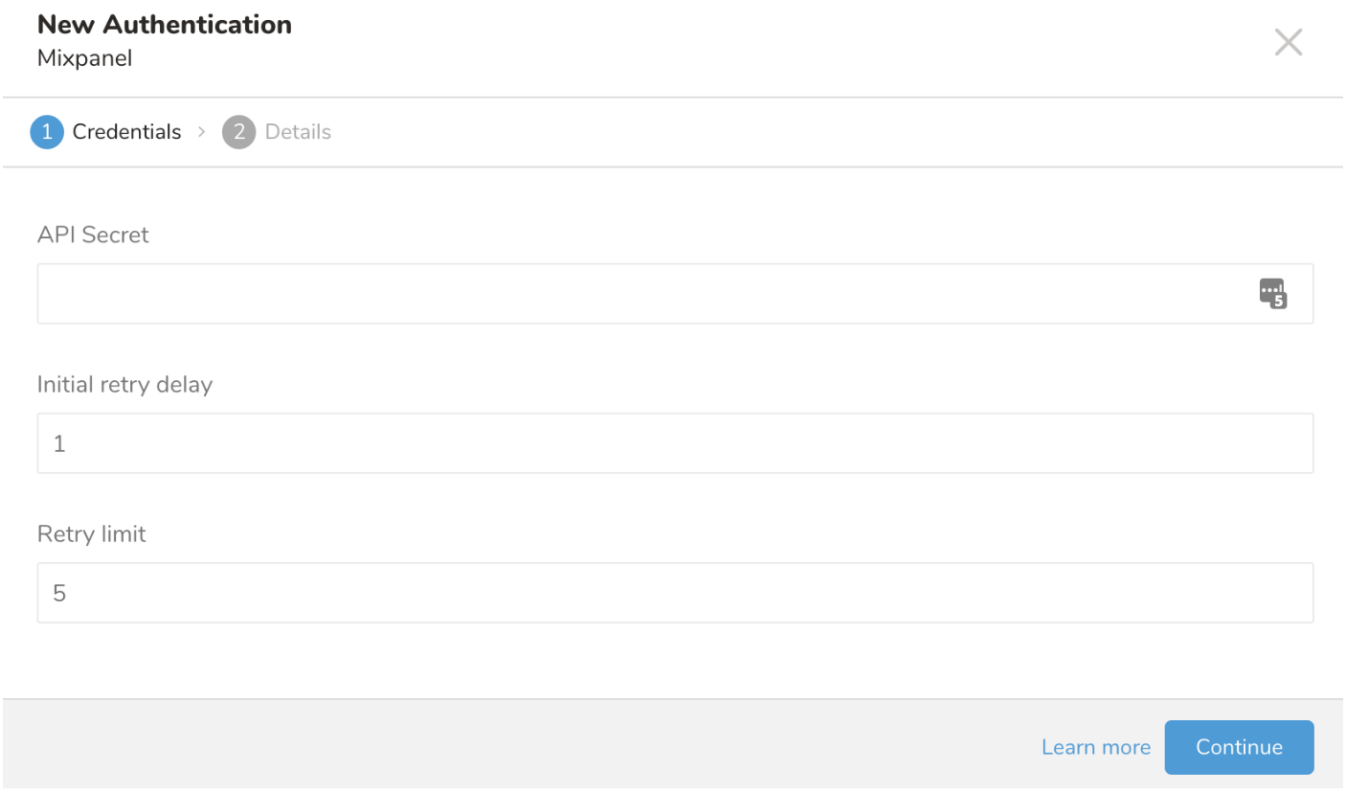
Use Case	Description
One-time migration	The organization is migrating away from Mixpanel and wants to keep a raw copy of the data
Incremental daily backup	The organization needs to have more granular access with SQL to event data inside Mixpanel

For sample workflows on how to import event data from Mixpanel, view Treasure Boxes.

Use TD Console

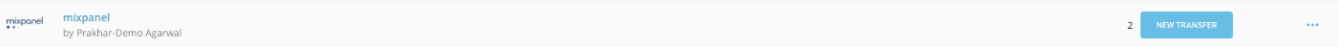
Create a New Connection

1. Navigate to Integrations Hub > Catalog
2. Search and select Mixpanel.
3. The following dialog opens. Edit the Mixpanel credential details.
4. Select **Continue** and provide a name for this connection.



Create a New Source

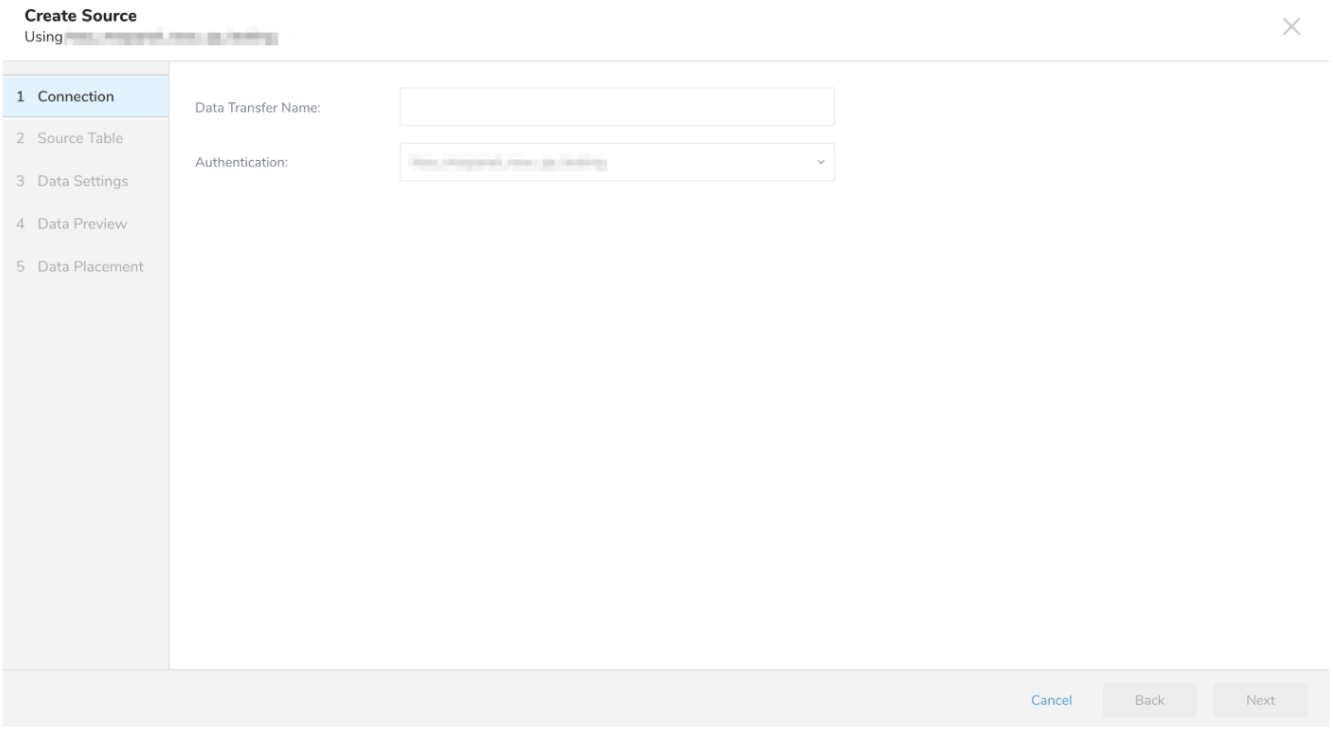
After creating the connection, you are automatically taken to the Authentications page. Look for the connection you created and select **New Source**.



Connection

The following dialog opens.

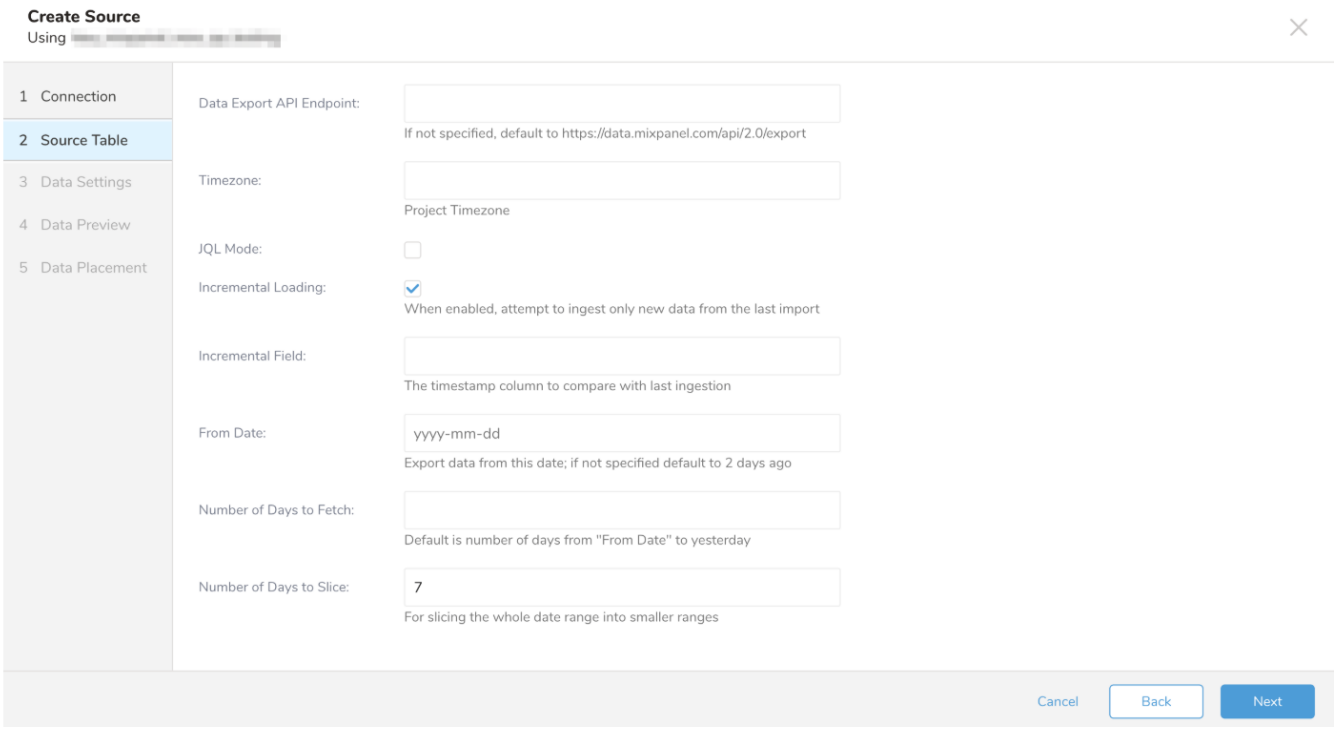
1. Name the data transfer and select **Next**.

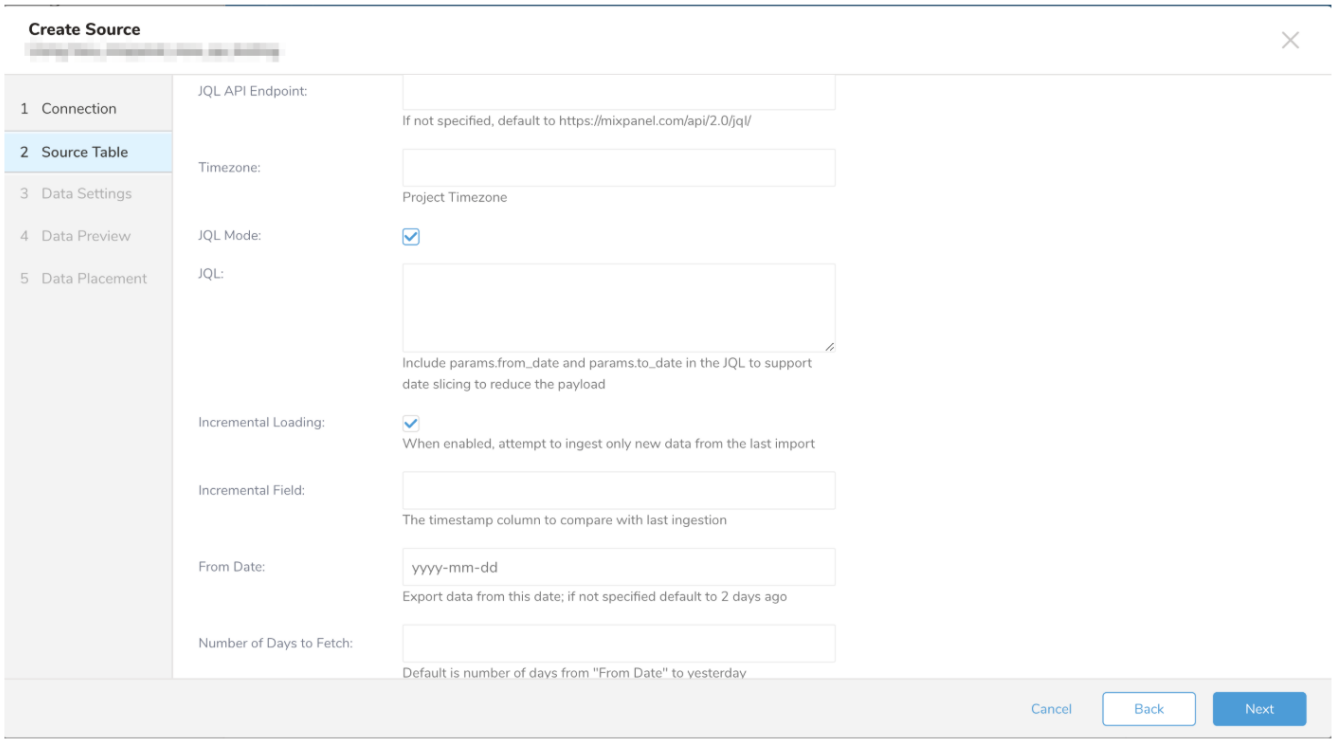


Source Table

The following dialog opens

1. Edit your source parameters and select **Next**.





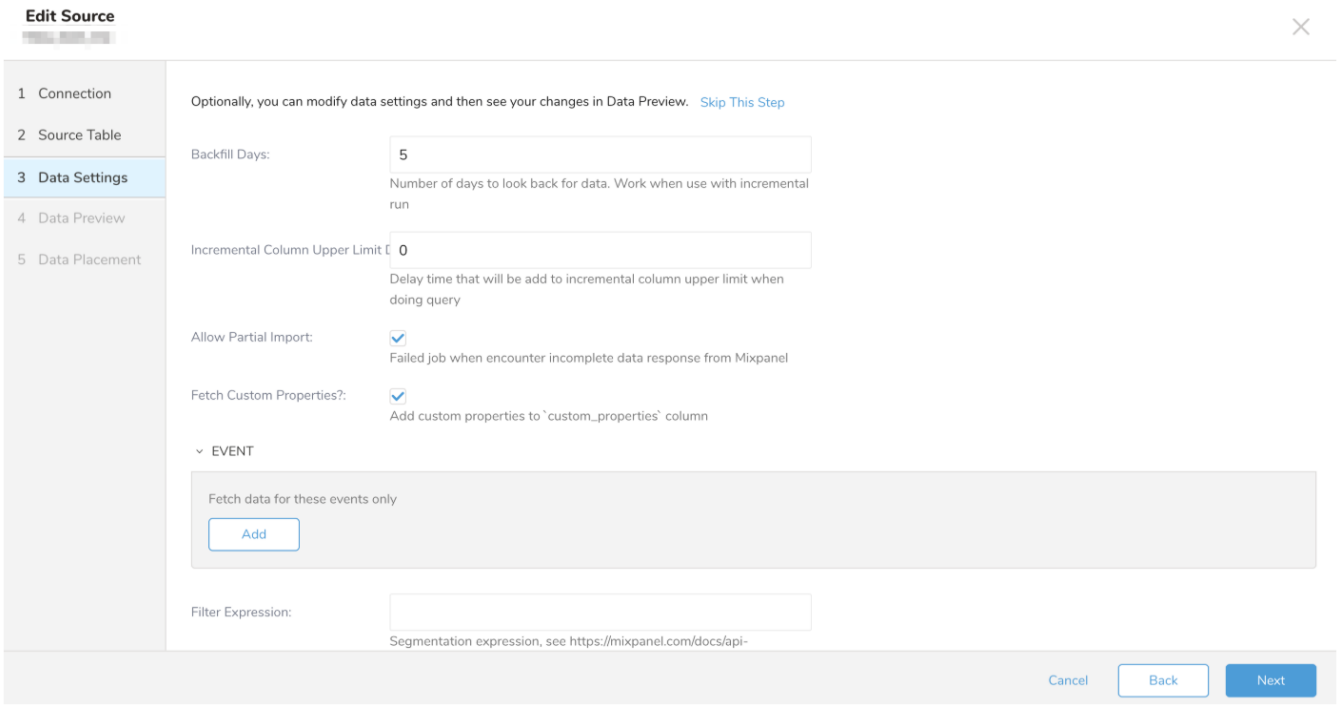
Parameters	Description
Data Export API Endpoint:	The endpoint you are using for the Export API call. If you leave it empty the default value is https://data.mixpanel.com/api/2.0/export/
JQL API Endpoint	The endpoint using for JQL API call. If you leave it empty the default value is https://mixpanel.com/api/2.0/jql/
Timezone	Your project timezone. It could be found in PROJECT SETTINGS >> YOUR PROJECT >> OVERVIEW.

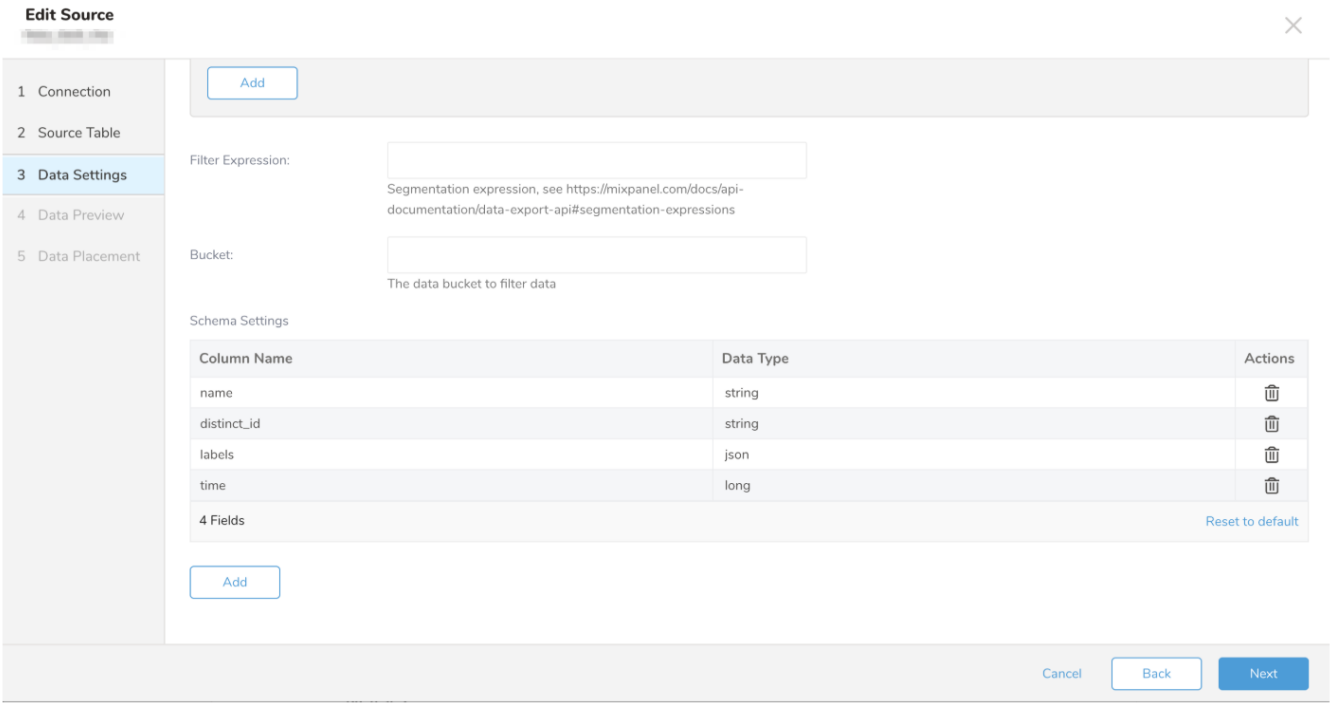
JQL Mode	Using JQL endpoint or just export endpoint. The default value is false
JQL	JQL script. It only effects when JQL Mode is true
Incremental Loading	Run your transfer in incremental mode or not
Incremental Field	which field is used as an index for incremental. The default value is `time`
From date	start date of the incremental
Number of Days to Fetch	total number of days to fetch data in one time
Number of Days to Slice	number of days that one request fetch

Data Settings

Your data settings can be edited on this page.

1. Edit the data settings or optionally, skip this page of the dialog.



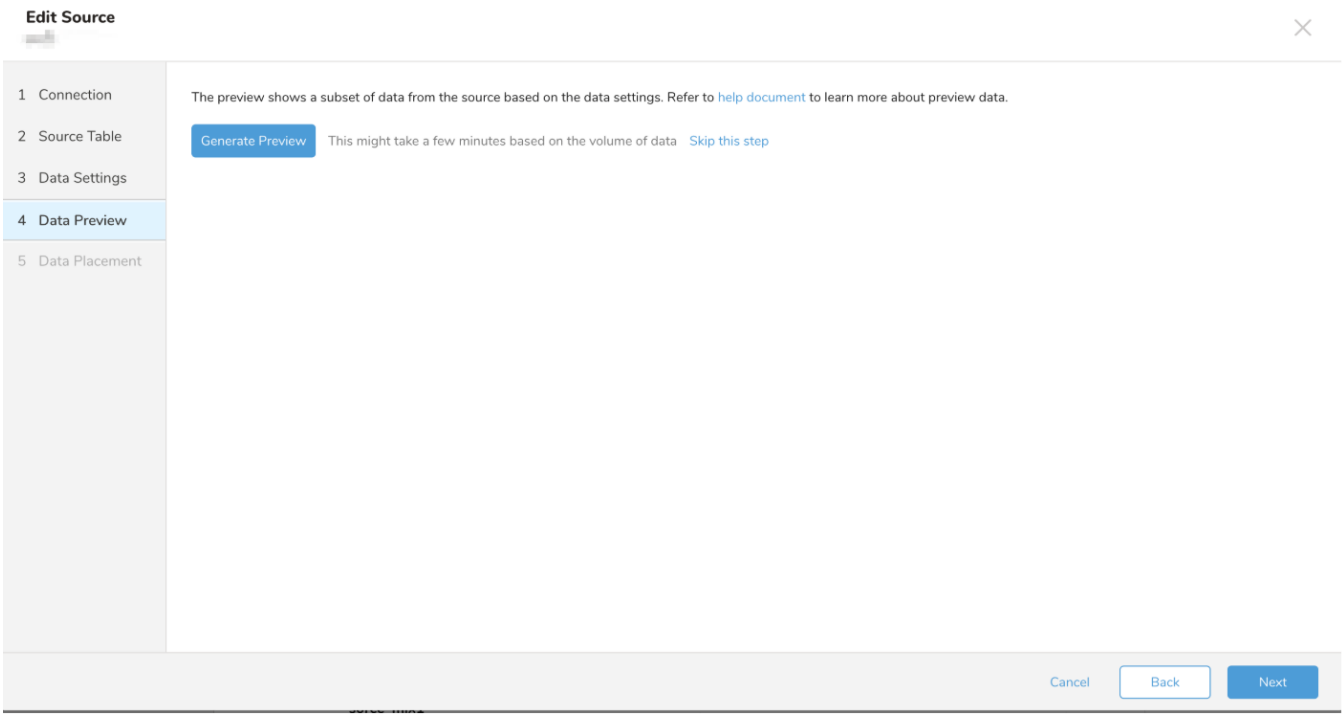


Parameters	Description
Backfill Days	The amount of time that will be subtracted from `from_date` to calculate the final `from_date` that is used for the API Request. This is due to the Mixpanel caching data on user devices before sending it to the Mixpanel server. It only affects when Incremental is true and Incremental Field is specified. The default value is 5
Incremental Column Upper Limit Delay In Seconds	The upper limit of the incremental column. When using export with the incremental column, the plugin will lock the upper limit of the incremental column query with the job start time. This is to support when Mixpanel has a delay in their processing. It only affects Export endpoint. The default value is 0.
Allow Partial Import	Allows the plugin to skip errors in the import. It only affects Export endpoint. The default value is true.
Fetch Custom Properties	Allows the plugin to import all custom properties for Export endpoint only. The default value is false.
Events	The events for filtering data for Export endpoint only
Filter Expression	The segmentation expression for Export endpoint only
Bucket	The data bucket to filter data for Export endpoint only
Schema	Your schema with columns names and types, which are stored in the TD table.

Data Preview

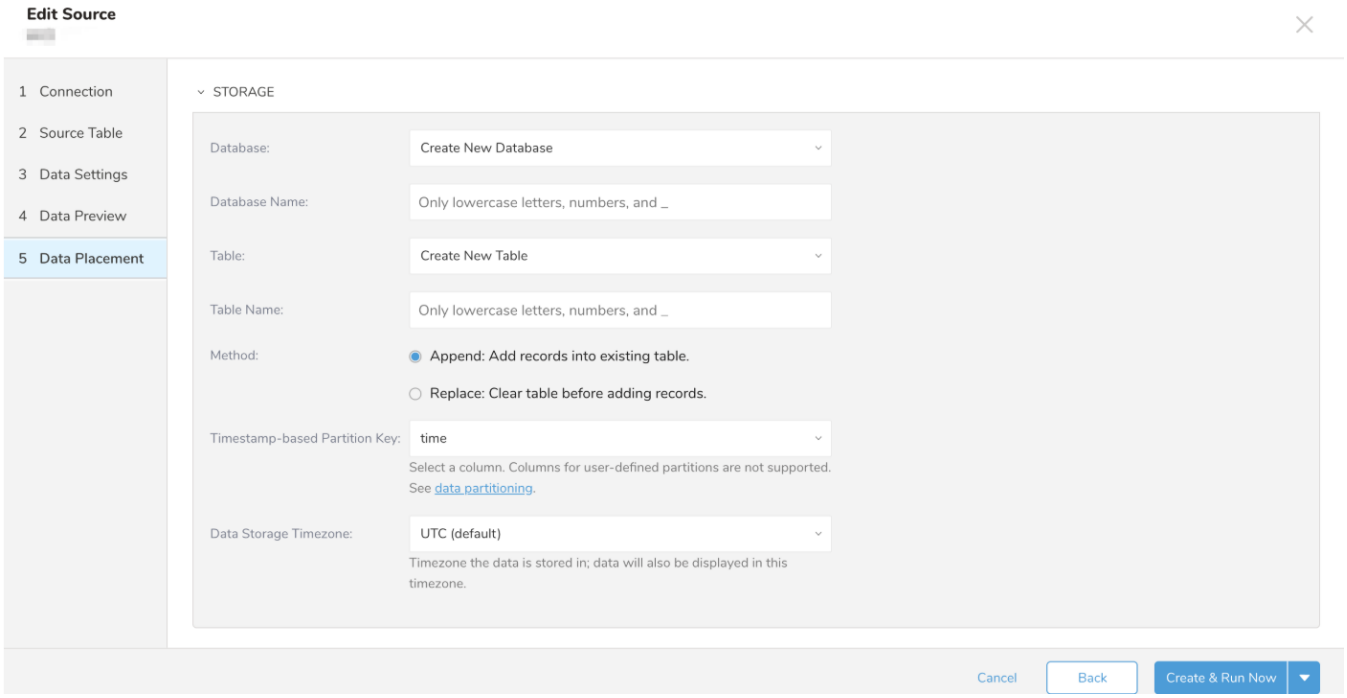
You can see a preview of your data in the Data Preview page before running the import by selecting Generate Preview. Data preview is optional and you can safely skip to the next page of the dialog if you choose to.

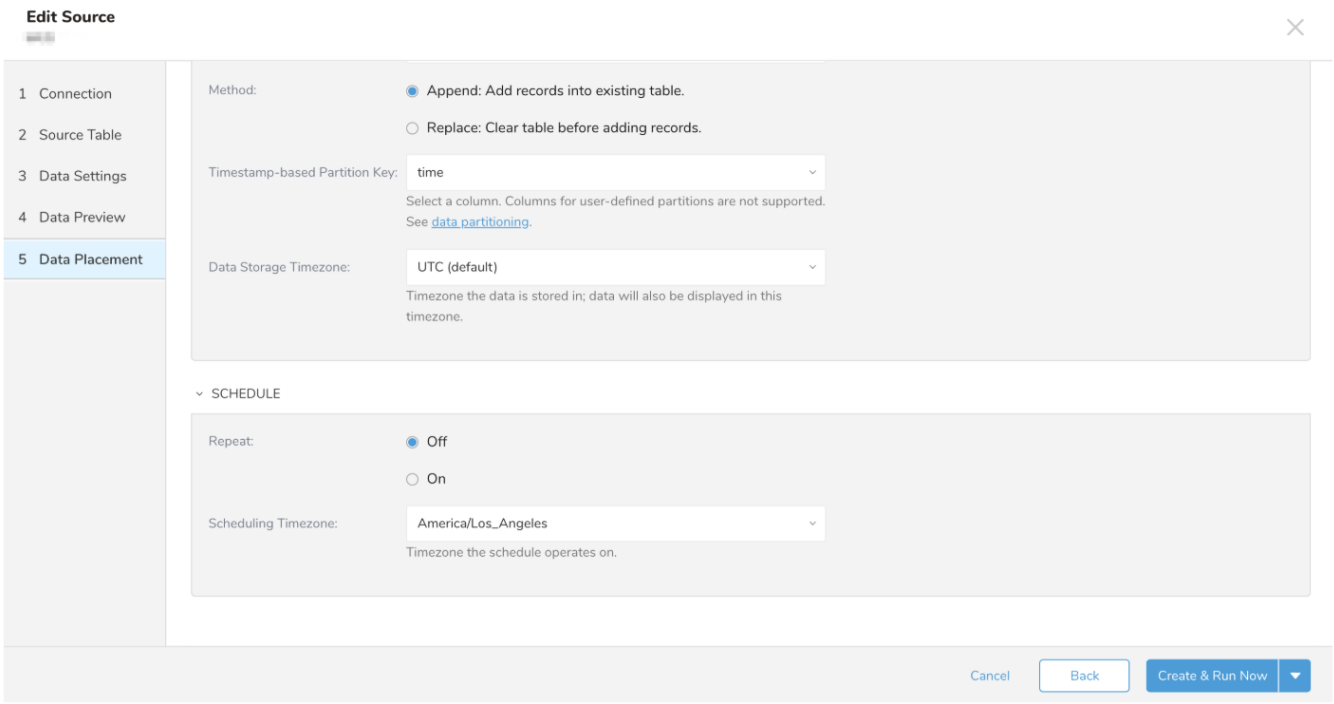
1. If you want to preview your data, select **Generate Preview.**
2. Verify the correct data is showing.
3. Select **Next**.



Data Placement

In this dialog, you will specify where your data will be placed and schedule how often it will run this import.





Under **Storage,** you will create a new or select an existing database and create a new or select an existing table for where you want to place the imported data.

1. Select a **Database** > **Select an existing** or **Create New Database**.
2. Select a **Table**> **Select an existing** or **Create New Table**.
3. Choose the Append or Replace 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.
    - **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.
4. 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.
5. Select the **Timezone** for your data storage.

Under **Schedule**, you can choose when and how often you want to run this query.

- Run once:
    1. Select **Off**.
    2. Select **Scheduling Timezone**.
    3. Select **Create & Run Now**.
- Repeat the query:
    1. Select **On**.
    2. Select the **Schedule**. The UI provides these four options: *@hourly*, *@daily* and *@monthly* or custom *cron*.
    3. You can also select **Delay Transfer** and add a delay of execution time.
    4. Select **Scheduling Timezone**.
    5. Select **Create & Run Now**.

After your transfer has run, you can see the results of your transfer in **Data Workbench** > **Databases** or **Jobs.**

Use Command Line

Install ‘td’ Command v0.11.9 or Later

Install the newest Treasure Data Toolbelt.

Look up Mixpanel API Credentials

Log in to your Mixpanel account, go to “Account” then “Projects” to look up your API key and API secret.

Create the Seed Config File (seed.yml)

Create a file called `seed.yml` as follows.

```
in:
  type: mixpanel
  api_key: MIXPANEL_API_KEY
  api_secret: MIXPANEL_API_SECRET
  timezone: 'UTC'
  from_date: "2015-10-28"
  fetch_days: 1
out:
  mode: append
```

This seed file is used to “guess” the full configuration file with column names. In this example, one day’s worth of data on Oct 28, 2015, is exported from Mixpanel to Treasure Data.

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

Guess Fields (Generate load.yml)

Based on `seed.yml`, generate `load.yml` with the `connector:guess` command.

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

Your column names and types depend on how you configure Mixpanel. The `load.yml` file should look as follows:

```
in:
  type: mixpanel
  api_key: MIXPANEL_API_KEY
  api_secret: MIXPANEL_API_SECRET
  timezone: UTC
  from_date: '2015-10-28'
  fetch_days: 1
  columns:
  - name: event
    type: string
  - name: "$browser"
    type: string
  - name: "$browser_version"
    type: long
  - name: "$city"
    type: string
  - name: "$current_url"
    type: string
  - name: "$initial_referrer"
    type: string
  - name: "$initial_referring_domain"
    type: string
  - name: "$lib_version"
    type: string
  - name: "$os"
    type: string
  - name: "$referrer"
    type: string
  - name: "$referring_domain"
    type: string
  - name: "$region"
    type: string
  - name: "$screen_height"
    type: long
```

```
  - name: "$screen_width"
    type: long
  - name: ENV
    type: string
  - name: RAILS_ENV
    type: string
  - name: distinct_id
    type: long
  - name: from
    type: string
  - name: job_id
    type: long
  - name: mp_country_code
    type: string
  - name: mp_lib
    type: string
  - name: name
    type: string
  - name: time
    type: long
filters:
- type: rename
  rules:
  - rule: upper_to_lower
  - rule: character_types
    pass_types: ["a-z", "0-9"]
    pass_characters: "_"
    replace: "_"
  - rule: first_character_types
    pass_types: ["a-z"]
    pass_characters: "_"
    prefix: "_"
  - rule: unique_number_suffix
    max_length: 128
out:
  mode: append
exec: {}
```

For more details on the `rename` filter, see rename filter plugin for Data Connector.

The Data Connector picked up fields such as:

- ENV
- RAILS_ENV
- $current_url

Column names are normalized in the type: rename filter section.

Preview Data Loading

Use `connector:preview` command to preview the data load.

```
$ td connector:preview load.yml
+------------------------------+----------------+----------------------+-------------
+-------------------------------------------------------------------------------
+-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
+----------------------------------+---------------------+------------
+-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
+-----------------------------+---------------+---------------------+--------------------+-------------
+--------------------------+-----------------+-----------------------------------+-------------
+------------------------+--------------+------------------------+------------+
| event:string                | _browser:string | _browser_version:long | _city:string | _current_url:
string                                                  | _initial_referrer:
string
| _initial_referring_domain:string | _lib_version:string | _os:string | _referrer:
string
| _referring_domain:string    | _region:string | _screen_height:long | _screen_width:long | env:string   |
rails_env:string          | distinct_id:long | from:string                     | job_id:long | mp_country_code:
string | mp_lib:string | name:string             | time:long  |
+------------------------------+----------------+----------------------+-------------
+-------------------------------------------------------------------------------
+-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
+----------------------------------+---------------------+------------
+-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
+-----------------------------+---------------+---------------------+--------------------+-------------
+--------------------------+-----------------+-----------------------------------+-------------
+------------------------+--------------+------------------------+------------+
| "view_job"                  | "Chrome"        | 46                    | "Tokyo"      | "https://console.
treasuredata.com/jobs/37608183/edit"                        |
"$direct"
| "$direct"                       | "2.7.1"             | "Mac OS X" | "https://console.treasuredata.com
/databases"
| "console.treasuredata.com"  | "Tky"        | 1440                | 2560               | "production" |
"production_heroku"       | 5421             | nil                               | 37608204    |
"JP"                  | "web"         | nil                     | 1445990400 |
| "view_job"                  | "Chrome"        | 46                    | nil          | "https://console.
treasuredata.com/query_forms/new"                           |
"$direct"
| "$direct"                       | "2.7.1"             | "Mac OS X" | "https://console.treasuredata.com/jobs
/37602729"
| "console.treasuredata.com"  | nil            | 800                 | 1280               | "production" |
"production_heroku"       | 6679             | nil                               | 37608227    |
"JP"                  | "web"         | nil                     | 1445990404 |
```

If the data looks good, then create the database and the table on Treasure Data where you wish to import Mixpanel data:

```
td db:create mixpanel_historical
td table:create mixpanel_historical app_name
```

Load Data

Submit the load job. It may 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 because Treasure Data's storage is partitioned by time (see architecture) 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.

```
td connector:issue load.yml --database mixpanel_historical --table app_name --time-column time
```

The preceding 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 TD this 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
```

You can assign Time Format column to the "Partitioning Key" by "--time-column" option.

You can check that the data is loaded by running a SQL query directly against your data in Treasure Data:

```
td query -T presto -w -d mixpanel_historical 'SELECT COUNT(1) FROM app_name'
```

Scheduling Incremental Data Loading

Unless you are migrating off of Mixpanel completely, Mixpanel data must be incrementally loaded into Treasure Data regularly. The Data Connector's scheduling function comes in handy for this purpose.

After scheduling, the Mixpanel Data Connector's successive runs increment the `from_date` parameter by `fetch_days`. For example, if the initial run in `load.yml` was

```
from_date: '2015-10-28'
fetch_days: 1
```

Then, the next run is as follows:

```
from_data: '2015-10-29'
fetch_days: 1
```

You **do not** need to update `load.yml` after it is uploaded since `from_date` field is automatically updated on the server side.

Suppose you wish to schedule a daily upload. Then, make sure that the initial `from_date` is at least two days ago and set `fetch_days: 1` in `load.yml`. Then, the following command creates a daily job called "daily_mixpanel_import" which loads historical data to `mixpanel_historical.app_name` on Treasure Data every day.

```
$ td connector:create \
    daily_mixpanel_import \
    "10 0 * * *" \
    mixpanel_historical \
    app_name \
    load.yml \
    --time-column time # optional
```

The historical runs of the import can be seen with `td connector:history <name>`, e.g., `td connector:history daily_mixpanel_import`.

Incremental Data Loading With Incremental Column

Certain Mixpanel account has project set up with additional field added to indicate the time data get processed by Mixpanel. For example: `mp_processing_time_ms`

User can add an additional parameter to Mixpanel Input Plugin `incremental_column`. The max `incremental_column` value of a run session will store and use as a filter for next run, Example: `where incremental_column > previous_run_max_value`.

Example:

```
in:
  type: mixpanel
  api_key: MIXPANEL_API_KEY
  api_secret: MIXPANEL_API_SECRET
  timezone: 'UTC'
  from_date: "2015-10-28"
  incremental_column: mp_processing_time_ms
  fetch_days: 1
out:
  mode: append
```

Look back for data with *back_fill_days*

For devices data like phone, tablets… Mixpanel can keep data in user's device for a while before send them to Mixpanel Server. So data appear in query can be delayed up to 5 days. When incremental import data from Mixpanel, we can missed data that are cached in user devices. To solve this issue we can set `back_fill_days` parameter(Default to 5). Plugin looks back for a number of days(`from_date - back_fill_days`). Due to performance issue, this feature only work with incremental_column.

Split range query into smaller API queries with *slice_range*

For some cases, the data return could be too big, data return in 1 query could exceed Mixpanel limitation and cause job to failed. In that case, we can split the big range query into smaller one using `slice_range` configuration parameter. This parameter are optional and default to ***7***.

Example:

```
    in:
      type: mixpanel
      api_key: MIXPANEL_API_KEY
      api_secret: MIXPANEL_API_SECRET
      timezone: 'UTC'
      from_date: "2015-10-20"
      incremental_column: mp_processing_time_ms
      fetch_days: 6
      slice_range: 2
    out:
      mode: append
```

The above configuration produces a query with the following range: `[2015-10-20,2015-10-21],[2015-10-22,2015-10-23],[2015-10-24,2015-10-25]` instead of [2015-10-20,2015-10-25]

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. Manual schema changes made to the target table remains intact with this mode.

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

Incremental for JQL mode

Incremental for JQL based on the idea that we could send parameters along with the JQL script, so parameters `params.start_time` and `params.end_time` are required in JQL script when using incremental in JQL mode.

```
in:
  incremental: true
  jql_mode: true
  from_date: '2020-03-05'
  jql_mode: true
  fetch_days: 1
  back_fill_days: 5
  slice_range: 8
  jql_script: 'function main() { return Events({ from_date: params.from_date, to_date:
    params.to_date})}'
in:
 incremental: true
 jql_mode: true
 from_date: '2020-03-05'
 jql_mode: true
 fetch_days: 10
 back_fill_days: 5
 slice_range: 8
 jql_script: 'function main() {return People().filter(function(user){return user.properties.signup_date >= new
Date("2016-01-01") &&
           user.properties.signup_date < new Date("2017-01-01");});}'
```