Creating a YAML Incremental Data Transfer File

You can configure incremental data transfers as part of managing your data pipeline. Use TD Workflows to ensure query processing steps in Treasure Data run only after the necessary data has been ingested into your account.

Importing the most recently added or modified records using an incremental ingest make the most sense when:

- Tables are too large to regularly re-import the entire table
  (for example, from large production databases)
- You are running frequent imports of updated data to keep the data as fresh as possible
  (for example, every 15 minutes)
- You want to minimize the number of rows ingested, to make the most efficient use of your Treasure Data account plan's capacity

Prerequisites

- Knowledge of Treasure Data, including the TD Toolbelt.
- Basic understanding of the Digdag scheduled execution and session_time.
- Ability to create a data connection and input transfer.
- An existing workflow for which you want to include an incremental data transfer.

Configuring your Unique ID Incremental Data Transfer

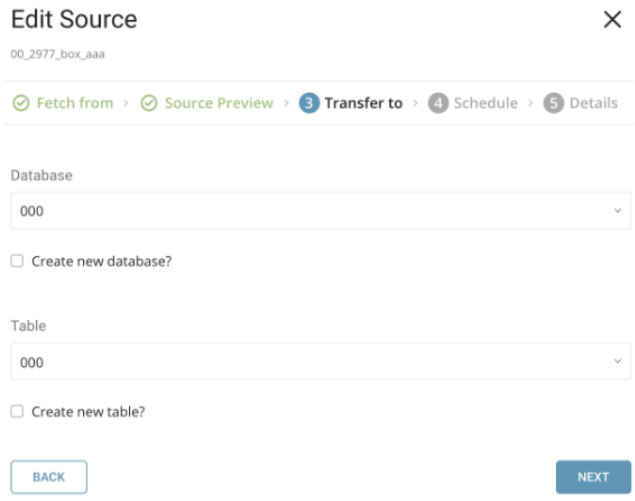
You can use the Unique ID associated with your Source to configure incremental data transfers.

When you are handling incremental processing logic from IDs or file names, such as using column names. Or, for example, when you want to keep track of the ID of the last record, and then pull from the next record onward at the next ingestion event.

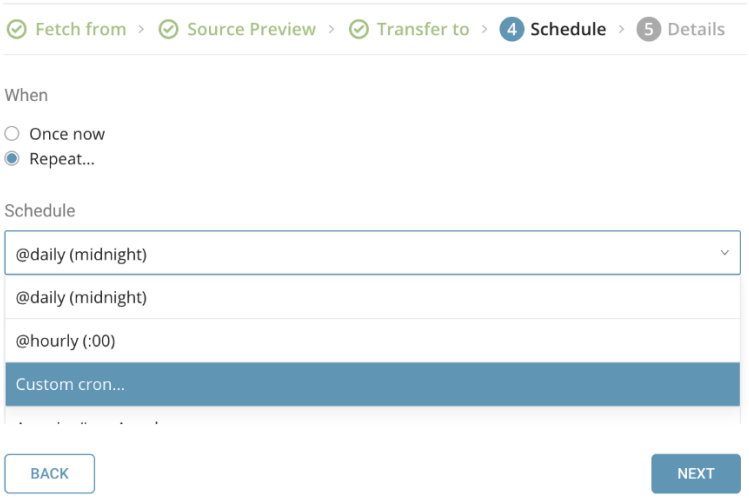
When you are not ingesting data on a regular basis.

When you want the system to be more self-healing. For example, when you have a single failure event. With self-healing, if an ingest fails, but the next succeeds, the 'diff' calculated will be from the last successful ingestion occurrence.

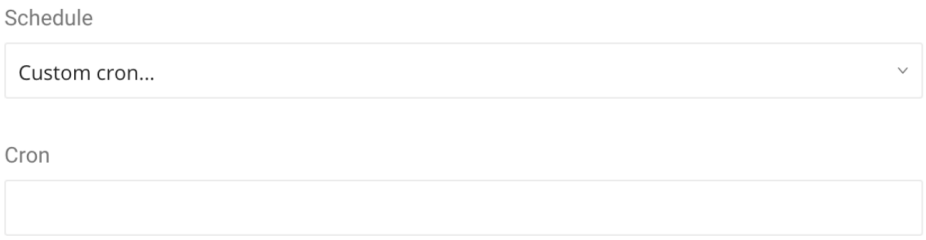
1. Navigate to Integrations > Sources.
2. Select the integration for which you want to define incremental data transfer.
3. Edit the Source.
4. Click Next to get to the Transfer To panel.



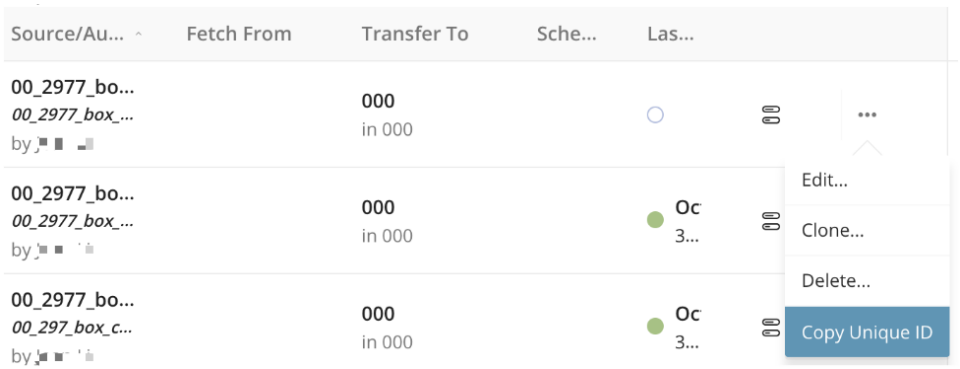
5. Save the name of the database and table. For example 000 and 000.
6. Click Next to get to the Schedule panel.
7. Select Repeat.
8. From the Schedule list, select Custom cron.



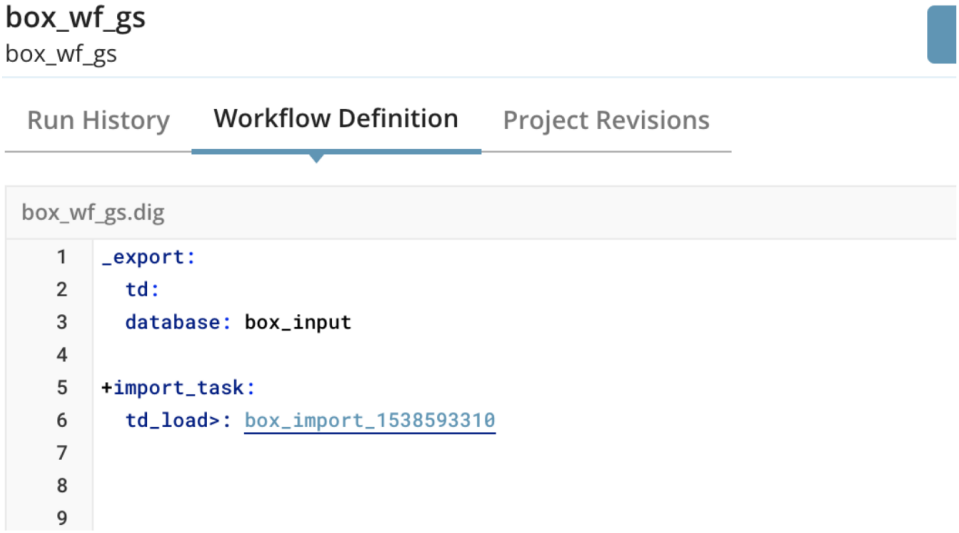
9. Delete the value from the Cron field. For example:



10. Navigate to Integrations > Sources.
11. For the integration from which you want to define incremental data transfer, select the more-ellipse ("…").



12. Select Copy Unique ID.
    The value is copied to the clipboard. For example, box_import_1540765752.
13. Navigate to Data Workbench > Workflows.
14. Select the workflow for which you want to define an incremental data transfer.
15. Open the workflow definition.
16. Add the td_load>: command with the Unique ID that you have saved to the clipboard.



17. Schedule your workflow. The incremental data load occurs every time you run that workflow.

Configuring a YAML Incremental Data Transfer

You can use this method of incremental data transfer when the incremental flow is based on a date.

In this partial example, the data connection configuration file is daily_load.yml. The database is `td_sample_db` and the table is `td_sample_table`. You'll need to refer to the data connection configuration file name in the workflow.

Confirm that creating a custom YAML file is the best approach for you to use for incremental data transfer.

The TD Toolbelt is required.

1. Create and run a workflow with an empty string ("") in the —cron option of the command. For example:

```
$ td connector:create daily_salesforce_marketing_cloud_import "" td_sample_db td_sample_table
daily_load.yml
```

2. Write down the name you gave this data transfer. For example, `daily_salesforce_marketing_cloud_import`.
3. Open the workflow in which you want the incremental transfer to run.
4. Add the `td_load>:` command with the data transfer name. For example:

```
+load: td_load>: config/daily_load.yml database: ${td.td_sample_db} table: ${td.td_sample_table}
+data_import: td_load>:daily_load_1491533104
```

5. Add a variable for last_session_time. For example:

```
timezone: UTC
schedule:
  daily>: 07:00:00
_export:
  td:
    dest_db: td_sample_db
    dest_table: td_sample_table
  salesforce:
    start_time: "${last_session_time}"
+prepare_table:
  td_ddl>:
    create_databases: ["${td.td_sample_db}"]
    create_tables: ["${td.td_sample_table}"]
+data_import:
  td_load>: config/daily_load.yml
  database: ${td.td_sample_db}
  table: ${td.td_sample_table}
```

6. Edit your `config/seed.yml` to use the last_session_time variable. For example:

```
in:
    type: salesforce
    login_url: https://<YOUR_DOMAIN_NAME>.salesforce.com
    auth_method: token
    username: <YOUR_EMAIL_ADDRESS>
    token: <YOUR_API_TOKEN>
    target: tickets
    start_time: ${salesforce.start_time} #use workflow variable
  out:
    mode: append
```

7. Schedule your workflow. The workflow runs a data load and fetches data incrementally since the last_session_time.