Avro Parser Function

The Avro parser plugin for Treasure Data's integrations parses files containing Avro binary data. The following Treasure Data integrations support the Avro parser:

- SFTP (version 1)
- Amazon S3
- Box
- OneDrive
- Microsoft Azure
- FTP

The Treasure Data Avro parser supports the following compression codecs:

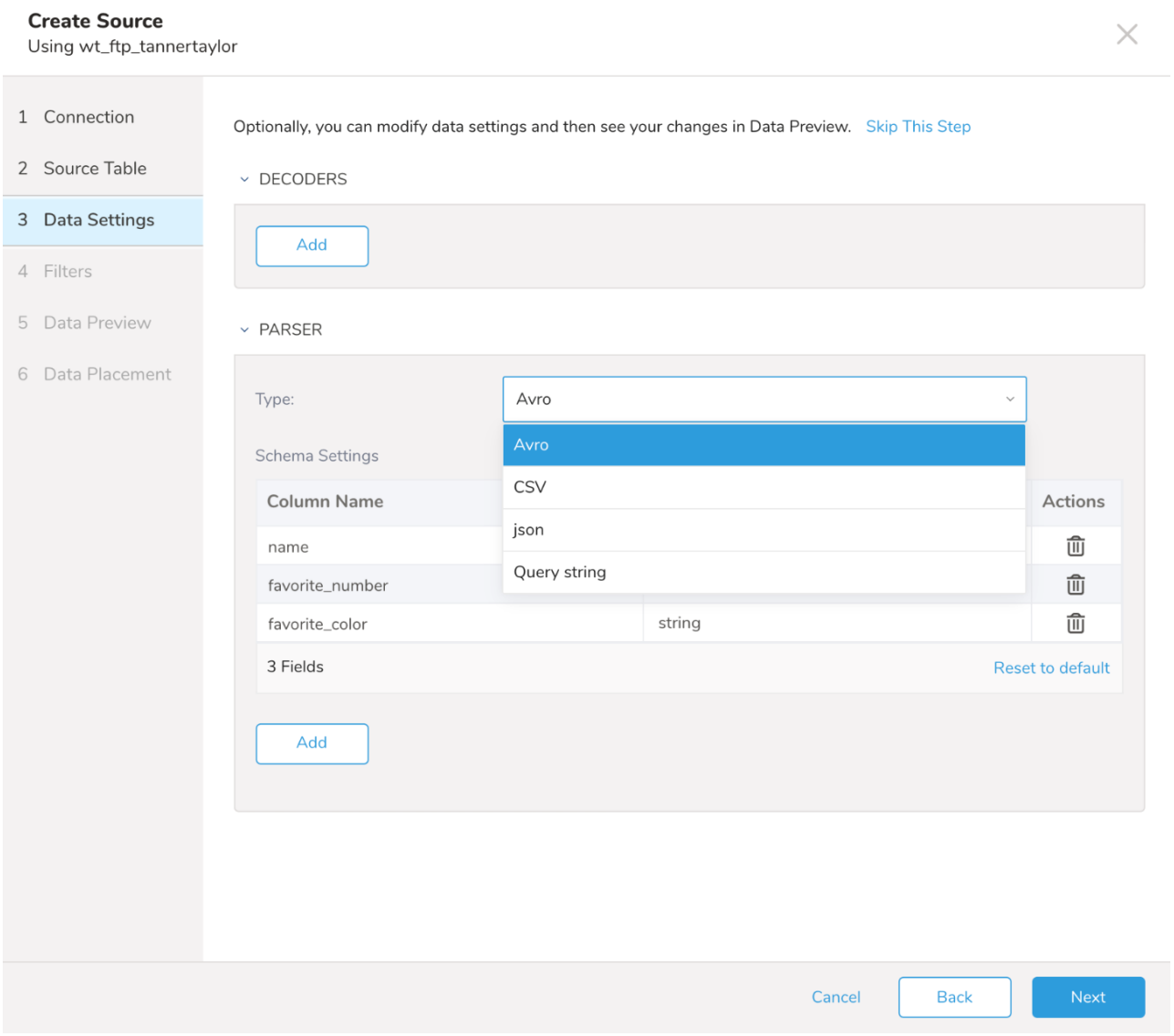
- Deflate
- Snappy

The parser is available for use from either the Treasure Data Console or the Treasure Data CLI.

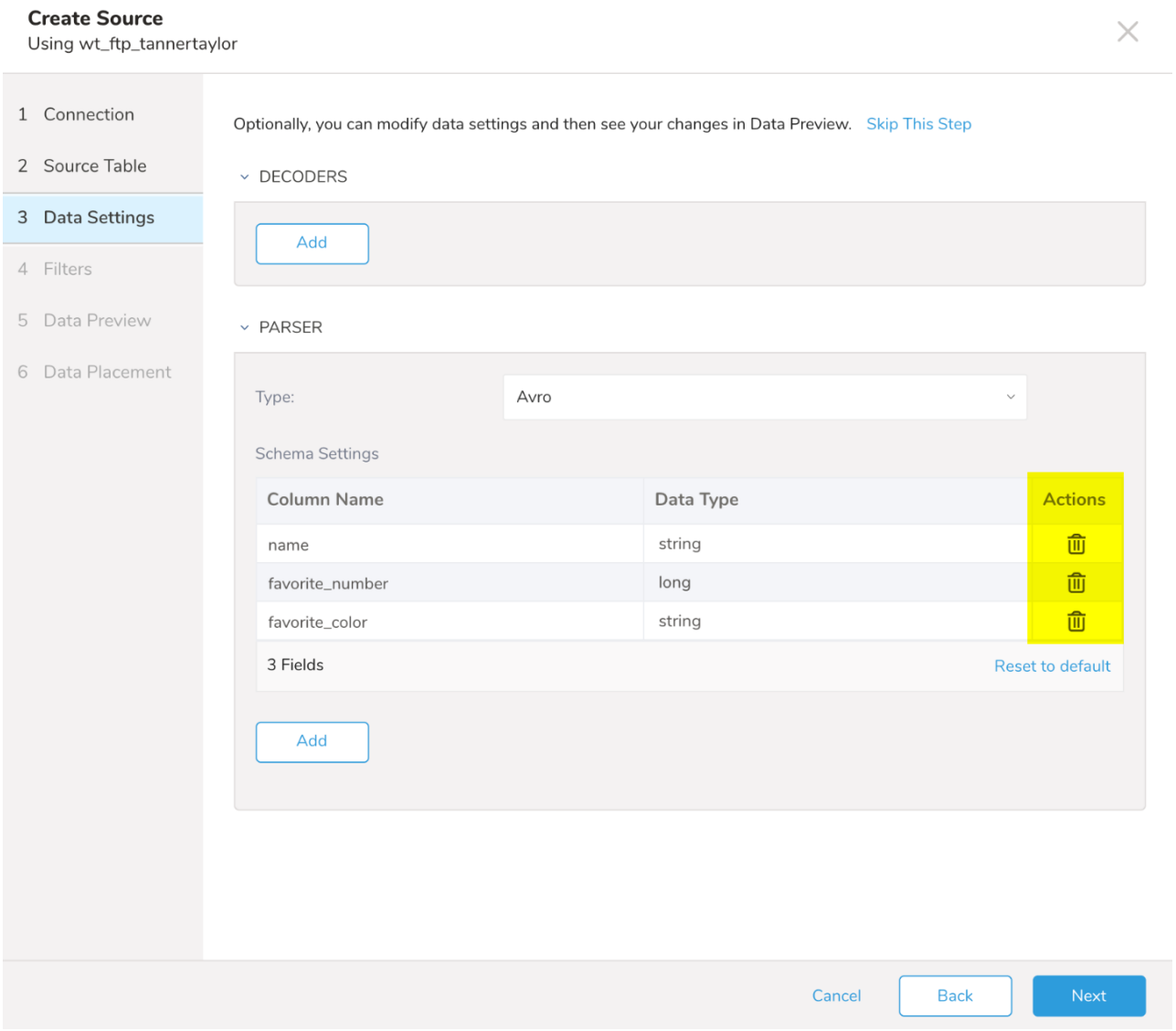
Using the Avro Parser Function from the Treasure Data Console

To use the Avro parser function in the Treasure Data console you will need to create an authentication for one of the supported integrations. Afterwards, you can use that authentication to create a source. In the process of creating the source, you will have the opportunity to either adjust or preview the data in the Avro file.

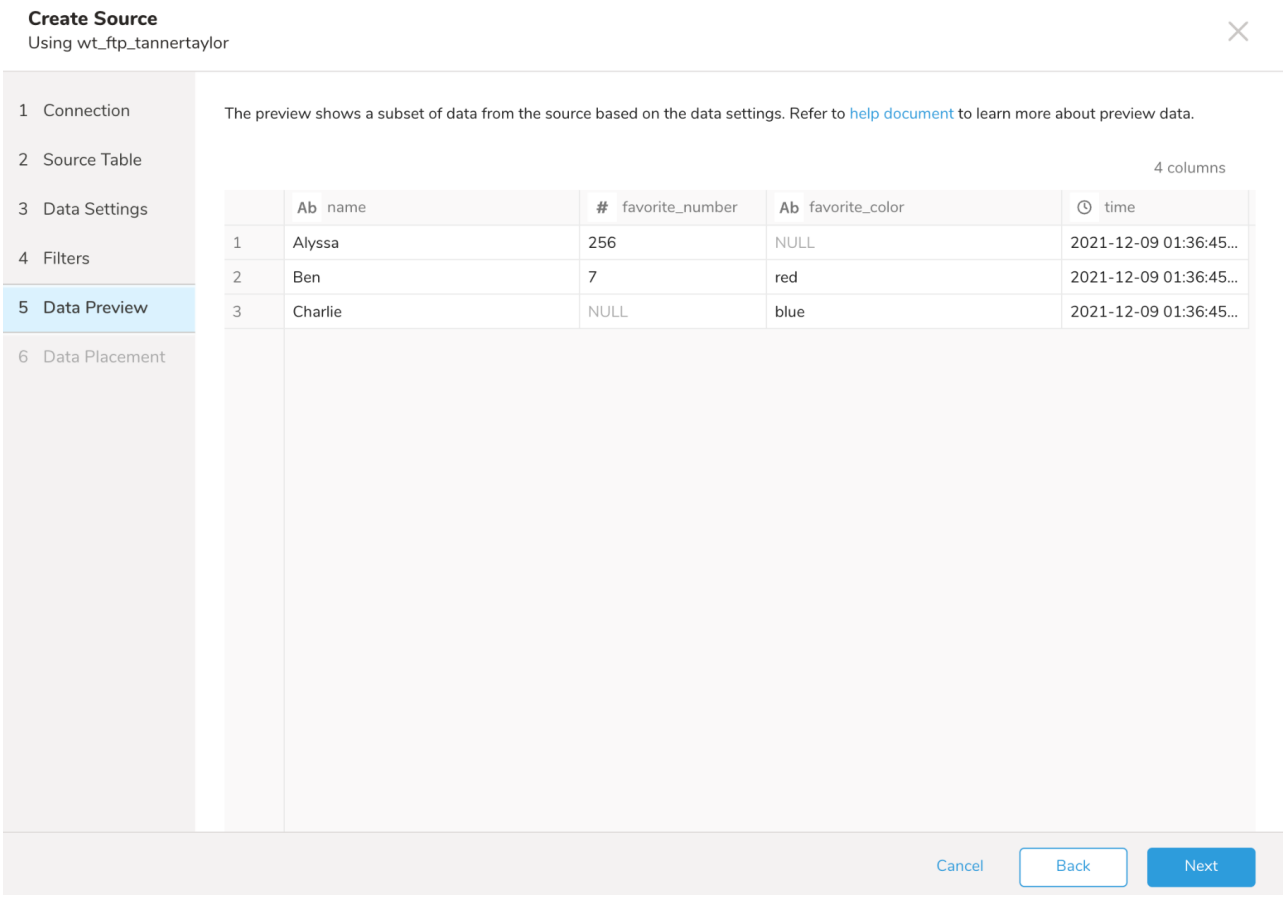
In step 3 of the Create Source interface, Data Settings, the TD Console should automatically select the Avro parser. If it does not, you can manually select it from the **Parser > Type** drop-down menu.



You also have the option to use the delete icon to prevent specific rows from being imported.



In step 5 of the Create Source interface, Data Preview, the TD Console displays a preview of the data.



Using the Avro Parser Function from the Treasure Data CLI

Using the Treasure Data CLI , you can import Avro data from the command line using the td connector command. This command takes a configuration file in `YAML` format as input. For example, using the sample file, users.avro, you could create a file named `importAvroFTP.yml` to import the sample file from an FTP site. The configuration file might look something like this:

```
in:
  type: ftp
  host: 10.100.100.10
  port: 21
  user: user1
  password: "password123"
  path_prefix: /misc/users.avro
  parser:
    type: avro
    columns:
    - {name: name, type: string}
    - {name: favorite_number, type: long}
    - {name: favorite_color, type: string}
out:
  mode: append
```

Manually Defining the Columns for the Avro Import

The **`columns`** section of the file is where you can define the schema of the Avro file by specifying key value pairs for `name`, `type`, and (in the case of a timestamp) format.

Key Value Pairs for Column Array

Column	Description

`name`	Name of the column
`type`	Type of the column • **boolean**: true or false • **long**: 64-bit signed integers • **double**: 64-bit floating-point numbers • **string** • **timestamp**: Date and time with nano-seconds precision • **json**
`format`	This option is only valid if the type of the column is timestamp.

Here are examples of how columns can be defined:

```
- {name: first_name, type: string}
- {name: favorite_number, type: long}
- {name: last_access, type: timestamp, format: '%Y-%m-%d %H:%M:%S.%N'}
```

After you have set up your configuration file, use the `td connector` command to perform the import. Here is an example of how that might look.

The following example assumes that the users.avro file is available on your FTP site. Additionally, for the manual import process to work, a time column is added to the importAvroFTP.yml file.

```
$  cat importAvroFTP.yml
in:
  type: ftp
  host: 10.100.100.10
  port: 21
  user: user1
  password: "password123"
  path_prefix: /misc/users.avro
  parser:
    type: avro
    columns:
    - {name: name, type: string}
    - {name: favorite_number, type: long}
    - {name: favorite_color, type: string}
    - {name: time, type: timestamp}
out:
  mode: append


$  td connector:preview importAvroFTP.yml
+-------------+----------------------+-----------------------+----------------+
| name:string | favorite_number:long | favorite_color:string | time:timestamp |
+-------------+----------------------+-----------------------+----------------+
| "Alyssa"    | 256                  | nil                   | nil            |
| "Ben"       | 7                    | "red"                 | nil            |
| "Charlie"   | nil                  | "blue"                | nil            |
+-------------+----------------------+-----------------------+----------------+
3 rows in set
Update importAvroFTP.yml and use 'td connector:preview importAvroFTP.yml' to preview again.
Use 'td connector:issue importAvroFTP.yml' to run Server-side bulk load.


$  td connector:issue importAvroFTP.yml --database wt_avro_db --table wt_avro_table --auto-create-table
Job 1228834439 is queued.
Use 'td job:show 1228834439' to show the status.

$  td job:show 1228834439
JobID       : 1228834439
Status      : success
Type        : bulkload
Database    : wt_avro_db
Config      :
```

```
---
embulk_config:
  in:
    host: 10.100.100.10
    port: 21
    user: user1
    password: "***"
    path_prefix: "/misc/users.avro"
    parser:
      type: avro
      columns:
      - name: name
        type: string
      - name: favorite_number
        type: long
      - name: favorite_color
        type: string
      - name: time
        type: timestamp
    type: ftp
  filters: []
  exec: {}
  out:
    type: td_internal
    mode: append
    plazma_dataset: 3867/wt_avro_db/wt_avro_table_20211209_055610_96b62b1d
config_diff:
  out:
    unique_transaction_name: bulkload.1228834439
    encrypt_start_at: 1461726840
Use '-v' option to show detailed messages.


# To see the contents of the table, you can use a TD Query command. Because the output of the query command is
verbose, much of the response has been removed from the following example:

$  td query -d wt_avro_db -T presto -w 'select * from wt_avro_table'
Job 1228232601 is queued.
Use 'td job:show 1228232601' to show the status.
  started at 2021-12-09T06:20:42Z
  presto version: 350
  executing query: select * from wt_avro_table
 .
 .
 .
 .

  finished at 2021-12-09T06:20:43Z
Status      : success
Result      :
+---------+-----------------+----------------+------------
+
| name    | favorite_number | favorite_color | time       |
+---------+-----------------+----------------+------------+
| Alyssa  | 256             | null           | 1639029373 |
| Ben     | 7               | red            | 1639029373 |
| Charlie | null            | blue           | 1639029373 |
+---------+-----------------+----------------+------------+
3 rows in set
```

Allowing the Parser to Guess the Columns for the Avro Import

You also have the option to let the parser make a best guess at the columns and data types being used in any particular file. The parser makes it "guesses" based on the conversion table shown here.

Default Type Conversions from Avro to TD

Avro Type	TD Data type

String	String
Bytes	String
Fixed	String
Enum	String
Null	String
Int	Long
Long	Long
Float	Double
Double	Double
Boolean	Boolean
Map	JSON
Array	JSON
Record	JSON

The process for having the parser guess the column schema is to run the `td connector:guess` command on your config file before running the `td connector:issue` command. Here is an example of how that process might look.

The following example assumes that the users.avro file is available on the FTP site specified. Additionally, the `columns` section of the importAvroFTP.yml file is stripped out. During the guess process, the parser creates the `columns` section and performs some limited fix up on the file.

```
$  cat importAvroFTP.yml
in:
  type: ftp
  host: 10.100.100.10
  port: 21
  user: user1
  password: "password123"
  path_prefix: /misc/users.avro
  parser:
    type: avro
out:
  mode: append


$  td connector:guess importAvroFTP.yml -o guessed.yml
Guessed configuration:

---
in:
  type: ftp
  host: 10.100.100.10
  port: 21
  user: user1
  password: "password123"
  path_prefix: /misc/users.avro
  parser:
    type: avro
    charset: UTF-8
    newline: CR
    columns:
    - {name: name, type: string}
    - {name: favorite_number, type: long}
    - {name: favorite_color, type: string}
out: {mode: append}
exec: {}
filters:
- from_value: {mode: upload_time}
```

```
  to_column: {name: time}
  type: add_time

Created guessed.yml file.
Use 'td connector:preview guessed.yml' to see bulk load preview.


$  td connector:preview guessed.yml
+-------------+---------------------+----------------------+-------------------------------+
| name:string | favorite_number:long | favorite_color:string | time:timestamp                |
+-------------+---------------------+----------------------+-------------------------------+
| "Alyssa"    | 256                 | nil                  | "2021-12-09 04:43:46.214 UTC" |
| "Ben"       | 7                   | "red"                | "2021-12-09 04:43:46.214 UTC" |
| "Charlie"   | nil                 | "blue"               | "2021-12-09 04:43:46.214 UTC" |
+-------------+---------------------+----------------------+-------------------------------+
3 rows in set
Update guessed.yml and use 'td connector:preview guessed.yml' to preview again.
Use 'td connector:issue guessed.yml' to run Server-side bulk load.


$  td connector:issue guessed.yml --database wt_avro_db --table wt_avro_table --auto-create-table
Database 'wt_avro_db' is created.
Table 'wt_avro_db.wt_avro_table' is created.
Job 1228213253 is queued.
Use 'td job:show 1228213253' to show the status.


$  td job:show 1228213253
JobID       : 1228213253
Status      : success
Type        : bulkload
Database    : wt_avro_db
Config      :
---
embulk_config:
  in:
    host: 10.100.100.10
    port: 21
    user: user1
    password: "***"
    path_prefix: "/misc/users.avro"
    parser:
      charset: UTF-8
      newline: CR
      type: avro
      columns:
      - name: name
        type: string
      - name: favorite_number
        type: long
      - name: favorite_color
        type: string
    type: ftp
  filters:
  - from_value:
      mode: upload_time
    to_column:
      name: time
    type: add_time
  exec: {}
  out:
    type: td_internal
    mode: append
    plazma_dataset: 3867/wt_avro_db/wt_avro_table_20211209_055610_96b62b1d
config_diff:
  out:
    unique_transaction_name: bulkload.1228213253
    encrypt_start_at: 1461726840
Use '-v' option to show detailed messages.
```

```
# To see the contents of the table, you can use a TD Query command. Because the output of the query command is
verbose, much of the response has been removed from the following example:

$  td query -d wt_avro_db -T presto -w 'select * from wt_avro_table'
Job 1228232601 is queued.
Use 'td job:show 1228232601' to show the status.
  started at 2021-12-09T06:20:42Z
  presto version: 350
  executing query: select * from wt_avro_table
 .
 .
 .
 .

  finished at 2021-12-09T06:20:43Z
Status      : success
Result      :
+---------+-----------------+----------------+------------
+
| name    | favorite_number | favorite_color | time       |
+---------+-----------------+----------------+------------+
| Alyssa  | 256             | null           | 1639029373 |
| Ben     | 7               | red            | 1639029373 |
| Charlie | null            | blue           | 1639029373 |
+---------+-----------------+----------------+------------+
3 rows in set
```

Example Configurations for Importing Avro Data from S3

Example of an importAvroS3.yml File

```
in:
  type: s3
  access_key_id: <access key>
  secret_access_key: <secret access key>
  bucket: <bucket name>
  path_prefix: users.avro
  parser:
    type: avro
    columns:
    - {name: name, type: string}
    - {name: favorite_number, type: long}
    - {name: favorite_color, type: string}
out: {mode: append}
exec: {}
```

Example of Running td connector:guess on an Avro File Hosted on S3

```
$ td connector:guess config.yml -o load.yml
Ignoring debase-0.3.0.beta19 because its extensions are not built. Try: gem pristine debase --version 0.3.0.
beta19
Guessed configuration:

---
in:
  type: s3
  access_key_id: <access key>
  secret_access_key: <secret access key>
  bucket: <bucket name>
  path_prefix: users.avro
  parser:
    charset: UTF-8
    newline: CR
    type: avro
    columns:
    - {name: name, type: string}
    - {name: favorite_number, type: long}
    - {name: favorite_color, type: string}
out: {}
exec: {}
filters:
- from_value: {mode: upload_time}
  to_column: {name: time}
  type: add_time

Created load.yml file.
Use 'td connector:preview load.yml' to see bulk load preview.
```