
airflow-provider-xlsx Documentation

Release 1.0.1

airflow-provider-xlsx contributors

Mar 25, 2022

CONTENTS

- 1 System Requirements 3**
- 2 Installation 5**
 - 2.1 Links 5
- 3 API documentation 7**
 - 3.1 xlsx_provider.operators.from_xlsx_operator 7
 - 3.2 xlsx_provider.operators.operators.from_xlsx_query_operator 8
 - 3.3 xlsx_provider.operators.operators.to_xlsx_operator 9
 - 3.4 xlsx_provider.common 10
- 4 Table of Contents 13**
- Python Module Index 15**
- Index 17**

Apache Airflow operators for converting XLSX files from/to Parquet, CSV and JSON.

SYSTEM REQUIREMENTS

- **Airflow Versions**
 - 2.0.0 or newer

INSTALLATION

```
pip install airflow-provider-xlsx
```

2.1 Links

- *Apache Airflow* <<https://github.com/apache/airflow>>
- *Project home page (GitHub)* <<https://github.com/andreax79/airflow-provider-xlsx>>
- *Documentation (Read the Docs)* <<https://airflow-provider-xlsx.readthedocs.io/en/latest/>>
- *openpyxl*, library to read/write Excel 2010 xlsx/xlsm/xltx/xltm files <<https://foss.heptapod.net/openpyxl/openpyxl>>
- *lrd*, library for reading data and formatting information from Excel files in the historical .xls format <<https://github.com/python-excel/xlrd>>
- *Python library for Apache Arrow* <<https://github.com/apache/arrow/tree/master/python>>

API DOCUMENTATION

3.1 `xlsx_provider.operators.from_xlsx_operator`

```
class xlsx_provider.operators.from_xlsx_operator.FromXLSXOperator(source, target, worksheet=0,
                                                                skip_rows=0, limit=None,
                                                                drop_columns=None,
                                                                add_columns=None,
                                                                types=None,
                                                                column_names=None,
                                                                file_format='parquet',
                                                                csv_delimiter=',',
                                                                csv_header='lower',
                                                                float_format='%g',
                                                                nullable_int=False, *args,
                                                                **kwargs)
```

Convert an XLSX/XLS file into Parquet or CSV file

Read an XLSX or XLS file and convert it into Parquet, CSV, JSON, JSON Lines(one line per record) file.

Parameters

- **source** (*str*) – Source filename (XLSX or XLS, templated)
- **target** (*str*) – Target filename (templated)
- **worksheet** (*str or int*) – Worksheet title or number (zero-based, templated)
- **skip_rows** (*int*) – Number of input lines to skip (default: 0, templated)
- **limit** (*int*) – Row limit (default: None, templated)
- **drop_columns** (*list of str*) – List of columns to be dropped
- **add_columns** (*list of str or dictionary of string key/value pair*) – Columns to be added (dict or list column=value)
- **types** (*str or dictionary of string key/value pair*) – force Parquet column types (dict or list column='str', 'int64', 'double', 'datetime64[ns]')
- **column_names** (*list of str*) – force columns names (list)
- **file_format** (*str*) – Output file format (parquet, csv, json, jsonl)
- **csv_delimiter** (*str*) – CSV delimiter (default: ',')
- **csv_header** (*str*) – Convert CSV output header case ('lower', 'upper', 'skip')
- **float_format** (*str*) – Format string for floating point numbers (default '%g')

- **nullable_int** (*bool*) – nullable integer data type support

class FileFormat(*value*)

File format enumerator (parquet/csv/json/jsonl)

execute(*context*)

This is the main method to derive when creating an operator. Context is the same dictionary used as when rendering jinja templates.

Refer to `get_template_context` for more context.

write(*names, columns, datatypes*)

Write data to file

write_csv(*names, columns, datatypes*)

Write data to CSV file

write_json(*names, columns, datatypes*)

Write data to JSON file

write_jsonl(*names, columns, datatypes*)

Write data to JSON Lines file

write_parquet(*names, columns, datatypes*)

Write the results in parquet format

3.2 xlsx_provider.operators.operators.from_xlsx_query_operator

```
class xlsx_provider.operators.from_xlsx_query_operator.FromXLSXQueryOperator(source, target,  
                                                                            worksheet=0,  
                                                                            skip_rows=0,  
                                                                            types=None,  
                                                                            file_format='parquet',  
                                                                            csv_delimiter=',',  
                                                                            csv_header='lower',  
                                                                            query=None,  
                                                                            table_name='xls',  
                                                                            use_first_row_as_header=False,  
                                                                            nullable_int=False,  
                                                                            *args,  
                                                                            **kwargs)
```

Execute an SQL query an XLSX/XLS file and export the result into a Parquet or CSV file

This operators loads an XLSX or XLS file into an in-memory SQLite database, executes a query on the db and stores the result into a Parquet, CSV, JSON, JSON Lines(one line per record) file. The output columns names and types are determinated by the SQL query output.

Parameters

- **source** (*str*) – Source filename (XLSX or XLS, templated)
- **target** (*str*) – Target filename (templated)
- **worksheet** (*str or int*) – Worksheet title or number (zero-based, templated)
- **skip_rows** (*int*) – Number of input lines to skip (default: 0, templated)

- **types** (*str* or *dictionary of string key/value pair*) – force Parquet column types (dict or list column='str', 'int64', 'double', 'datetime64[ns]')
- **file_format** (*str*) – Output file format (parquet, csv, json, jsonl)
- **csv_delimiter** (*str*) – CSV delimiter (default: ',')
- **csv_header** (*str*) – Convert CSV output header case ('lower', 'upper', 'skip')
- **query** (*str*) – SQL query (templated)
- **table_name** – Table name (default: 'xls', templated)
- **use_first_row_as_header** (*bool*) – if true, use the first row as column names otherwise use A, B, C, ... as column names
- **nullable_int** (*bool*) – nullable integer data type support

class FileFormat(*value*)

File format enumerator (parquet/csv/json/jsonl)

execute(*context*)

This is the main method to derive when creating an operator. Context is the same dictionary used as when rendering jinja templates.

Refer to `get_template_context` for more context.

write(*result*)

Write data to file

write_csv(*result*)

Write data to CSV file

write_json(*result*)

Write data to JSON file

write_jsonl(*result*)

Write data to JSON Lines file

write_parquet(*result*)

Write the results in parquet format

3.3 xlsx_provider.operators.operators.to_xlsx_operator

class xlsx_provider.operators.to_xlsx_operator.ToXLSXOperator(*source, target, worksheet=0, skip_rows=0, csv_delimiter=',, *args, **kwargs*)

Convert Parquest, CSV, JSON, JSON Lines into XLSX

Read a Parquest, CSV, JSON, JSON Lines(one line per record) file and convert it into XLSX

Parameters

- **source** (*str*) – source filename (type is detected by the extension, templated)
- **target** (*str*) – target filename (templated)
- **csv_delimiter** (*str*) – CSV delimiter (default: ',')
- **skip_rows** (*int*) – Number of input lines to skip (default: 0, templated)

class FileFormat(*value*)

File format enumerator (parquet/csv/json/jsonl)

execute(*context*)

This is the main method to derive when creating an operator. Context is the same dictionary used as when rendering jinja templates.

Refer to `get_template_context` for more context.

3.4 xlsx_provider.common

`xlsx_provider.common.DEFAULT_CSV_DELIMITER = ','`
Default CSV delimiter

`xlsx_provider.common.DEFAULT_CSV_HEADER = 'lower'`
Default CSV header case

`xlsx_provider.common.DEFAULT_FLOAT_FORMAT = '%g'`
Default float format

`xlsx_provider.common.DEFAULT_FORMAT = 'parquet'`
Default output format

`xlsx_provider.common.DEFAULT_TABLE_NAME = 'xls'`
Default Query Operator table name

class `xlsx_provider.common.FileFormat`(*value*)
File format enumerator (parquet/csv/json/jsonl)

`xlsx_provider.common.INDEX_COLUMN_NAME = '_index'`
Index column name

`xlsx_provider.common.NUMERIC_TYPES = ('int64', 'Int64', 'double')`
Numeric data type

`xlsx_provider.common.TYPE_DATETIME = 'datetime64[ns]'`
Datetime data type

`xlsx_provider.common.TYPE_DOUBLE = 'double'`
Double data type

`xlsx_provider.common.TYPE_INT = 'int64'`
Integer data type

`xlsx_provider.common.TYPE_NULLABLE_INT = 'Int64'`
Integer data type with possibly missing value

`xlsx_provider.common.TYPE_STRING = 'str'`
String data type

`xlsx_provider.common.VERSION = '1.0.1'`
Plugin Version

`xlsx_provider.common.XLSX_EPOCH = datetime.datetime(1900, 1, 1, 0, 0)`
XLSX Epoch

`xlsx_provider.common.XLS_EPOCH = datetime.datetime(1899, 12, 30, 0, 0)`
XLS Epoch - see <https://support.microsoft.com/en-us/help/214326/excel-incorrectly-assumes-that-the-year-1900-is-a-leap-year>

`xlsx_provider.common.col_number_to_name`(*col_number*)
Convert a column number to name (e.g. 0 -> '_index', 0 -> A, 1 -> B)

Parameters `col_number` (*int*) – column number

`xlsx_provider.common.copy_cells(source, target)`

Copy cells from source worksheet to target

`xlsx_provider.common.get_column_names(sheet, skip_rows=0)`

Extract the column names from the first row of the worksheet

Parameters

- **sheet** (*Worksheet*) – worksheet
- **skip_rows** (*int*) – Number of input lines to skip

`xlsx_provider.common.prepare_value(name, value)`

Try cast string to int and float

`xlsx_provider.common.print_sheet(sheet, fileobj=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>)`

Print a sheet on standard output as CSV

`xlsx_provider.common.rmdiacritics(char)`

Return the base character without diacritics (eg. accents)

TABLE OF CONTENTS

- [genindex](#)
- [modindex](#)
- [search](#)

PYTHON MODULE INDEX

X

`xlsx_provider`, [7](#)
`xlsx_provider.common`s, [10](#)
`xlsx_provider.operators.from_xlsx_operator`, [7](#)

INDEX

C

`col_number_to_name()` (in module `xlsx_provider.common`s), 10
`copy_cells()` (in module `xlsx_provider.common`s), 10

D

`DEFAULT_CSV_DELIMITER` (in module `xlsx_provider.common`s), 10
`DEFAULT_CSV_HEADER` (in module `xlsx_provider.common`s), 10
`DEFAULT_FLOAT_FORMAT` (in module `xlsx_provider.common`s), 10
`DEFAULT_FORMAT` (in module `xlsx_provider.common`s), 10
`DEFAULT_TABLE_NAME` (in module `xlsx_provider.common`s), 10

E

`execute()` (`xlsx_provider.operators.from_xlsx_operator.FromXLSXOperator` method), 8

F

`FileFormat` (class in `xlsx_provider.common`s), 10
`FromXLSXOperator` (class in `xlsx_provider.operators.from_xlsx_operator`), 7
`FromXLSXOperator.FileFormat` (class in `xlsx_provider.operators.from_xlsx_operator`), 8

G

`get_column_names()` (in module `xlsx_provider.common`s), 11

I

`INDEX_COLUMN_NAME` (in module `xlsx_provider.common`s), 10

M

module
`xlsx_provider`, 7
`xlsx_provider.common`s, 10

`xlsx_provider.operators.from_xlsx_operator`, 7

N

`NUMERIC_TYPES` (in module `xlsx_provider.common`s), 10

P

`prepare_value()` (in module `xlsx_provider.common`s), 11
`print_sheet()` (in module `xlsx_provider.common`s), 11

R

`rmDiacritics()` (in module `xlsx_provider.common`s), 11

T

`TYPE_DATETIME` (in module `xlsx_provider.common`s), 10
`TYPE_DOUBLE` (in module `xlsx_provider.common`s), 10
`TYPE_INT` (in module `xlsx_provider.common`s), 10
`TYPE_NULLABLE_INT` (in module `xlsx_provider.common`s), 10
`TYPE_STRING` (in module `xlsx_provider.common`s), 10

V

`VERSION` (in module `xlsx_provider.common`s), 10

W

`write()` (`xlsx_provider.operators.from_xlsx_operator.FromXLSXOperator` method), 8
`write_csv()` (`xlsx_provider.operators.from_xlsx_operator.FromXLSXOperator` method), 8
`write_json()` (`xlsx_provider.operators.from_xlsx_operator.FromXLSXOperator` method), 8
`write_jsonl()` (`xlsx_provider.operators.from_xlsx_operator.FromXLSXOperator` method), 8
`write_parquet()` (`xlsx_provider.operators.from_xlsx_operator.FromXLSXOperator` method), 8

X

`XLS_EPOCH` (in module `xlsx_provider.common`s), 10
`XLSX_EPOCH` (in module `xlsx_provider.common`s), 10

- xlsx_provider
 - [module, 7](#)
- xlsx_provider.commonsmodule, [10](#)
- xlsx_provider.operators.from_xlsx_operator
 - [module, 7](#)