xl2hdb.py – Python based Excel Spreadsheet HDB Data Loader

# Overview

The xl2hdb.py Python data loader is designed for loading data from Excel spreadsheets to the HDB R tables. Data is written to the r\_base table with an ‘O’ in the overwrite\_flag field by default, with metadata values defined in the source data spreadsheets. It is located underneath the “python” subdirectory in the HDB repository and relies on the generic lib/hdb.py class, which provides database connection and access to HDB SQL packages like ts\_xfer for writing arrays of real data.

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure : Organization of python loader directory.

# Installation/Requirements

The following python installation and packages are required:

* Python 3.9
* Cx\_oracle
* Pandas
* Argparse

# Instructions

## Source Data Spreadsheet Format

There are two allowed source data spreadsheet formats. The first is for writing real data without data tags, and the second is for writing real data with data tags. The loader will automatically determine which format is being used. Both formats require sheet names that are mapped to HDB datatypes in the hdb\_ext\_data\_code table.

### Without Data Tags

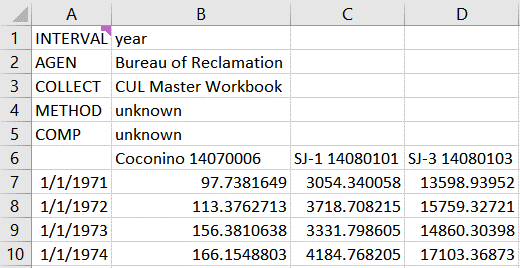


Figure : Excel source data spreadsheet format with generic site metadata.

The above screenshot is an example of a source data workbook formatted for use with xl2hdb.py. Column A defines r\_base.start\_date\_time for each data point, and the values in row 6 of the columns to the right must match unique values from hdb\_site.site\_name (case insensitive). Rows 1-5 determine metadata values for r\_base.interval, r\_base.agen\_id, r\_base.collection\_system\_id, r\_base.method\_id, and r\_base.computation\_id. In this case, the metadata values defined in cells B1:B5 apply to each site column but they can also be defined per site, as shown below. Also note the use of blank cells, so that time ranges for sites don’t have to match.

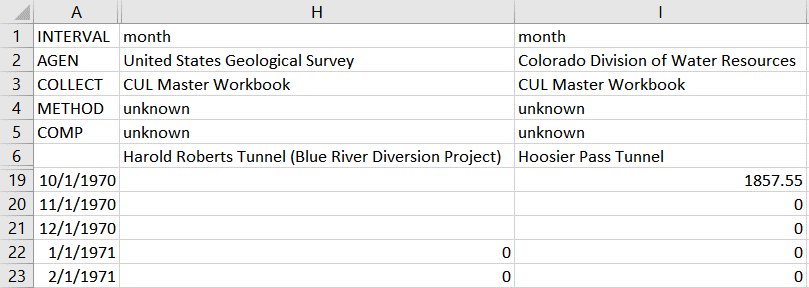


Figure : Excel source data workbook with site specific metadata

### With Data Tags

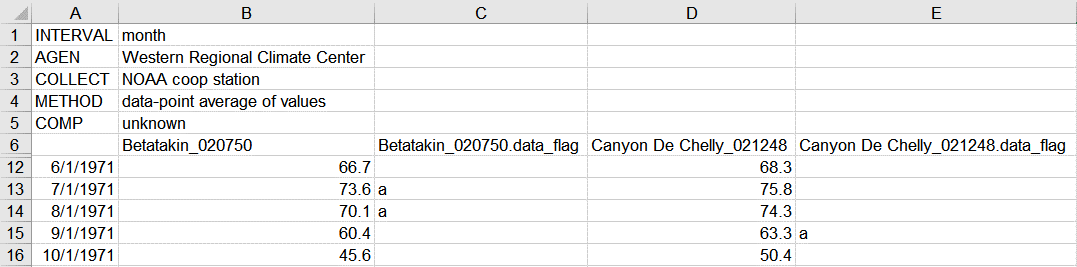


Figure : Excel source data workbook with generic site metada and data-point specific data flags.

The above screenshot is an example of a source data workbook that contains data point specific data flags. Each site column must be paired with a column named as <hdb\_site.site\_name>.data\_flag. The value in the data flag column will be written to r\_base.data\_flags, which is a 20-byte field. In the example above, Betatakin\_020750 will have its values for 7/1/1971 and 8/1/1971 written with an ‘a’ in the data\_flags field, while the other values will be written with NULL in the data\_flags field.

## Connection File

A connection file is used to supply database connection credentials. If stored in the hdb/python directory, use \*.auth to ensure that the file is ignored by Git. The screenshot below shows required connection file format.

Text

Description automatically generated

Figure : Database connection file format.

## Script Arguments

After preparing a source data spreadsheet, run the loader with “python xl2hdb.py” and the following arguments. Additionally, “python xl2hdb.py -h” will display similar help information on the command line while using the loader.

* Required Arguments
  + -a/--authFile – path to database connection file
  + -f/--file – path to source Excel workbook
  + -s/--sheet – sheet name to load from source Excel workbook
* Optional Arguments
  + -c/--column – Name of a single site column to load. If not provided, all site columns will be loaded
  + -l/--loadingApp – Loading application name, defaults to xl2hdb.py
  + -d/--dataCodeSys – Data code system name from hdb\_ext\_data\_code\_sys, which is used to map Excel sheet names to HDB datatypes. Defaults to “CUL Sheet Names”
* Optional switches
  + --verbose - Print additional detail to command line as the process is executed
  + --nooverwrite - Do not write an ‘O’ to the r\_base.overwrite\_flag field