Documentation for Python scripts to generate Workbench files

Paul Sutherland, last update 2025/1

Requirements

- Python. I made this on 3.11, not sure which one is necessary.
- Python libraries, installed by running: python -m pip install pandas[all] edtf-validate moviepy mutagen

File usage - main workflow

In order, these work as one workflow. Nodes_from_WB.py is useful but could be skipped if you don't need it.

Create_Fillable.py

Prepopulates a "Fillable" Excel spreadsheet that uses simplified column headers and formatting requirements compared to Workbench. Uses file information to fill a few fields. Optionally (recommended) uses exported ArchivesSpace data to fill other fields with archival metadata.

If using, you might want to keep this ArchivesSpace file for adding Digital Object nodes.

*Limitation: There's currently no way to use the ArchivesSpace data but **without** having the files also present. This could be an option to add to this file, but it would require a fairly large rewrite of the logic. If you want solely the headers into a spreadsheet, just copy them from _headers.*

Preparation for use:

- Copy all files/folders of files to be uploaded into "files_to_upload"
- Rename files so that file and folder names conform to Workbench standards laid out in APS Github
- If using ArchivesSpace data:
    - Go to collection in ArchivesSpace and generate a Bulk Update Spreadsheet, selecting records and leaving all column types checked
    - Edit this file (unprotect the sheet):
        - Leave the first two rows (human-readable header and machine-readable header) alone
        - Delete other rows so that you are left only with the exact records you wish to prepopulate. Any difference between the number of records here and the number of files/folders in "files_to_upload" will cause an error.
    - Put this ArchivesSpace spreadsheet in "metadata" and name it something usable

    - Note: The script will assume that the order of the ArchivesSpace spreadsheet matches the alphabetical order of files/folders in “files_to_upload”

Run the command line by opening this directory in terminal/CD’ing to directory.

Example command line: python Create_Fillable.py book cnair_book --AS haas_s9.xlsx

Command line arguments:

- “book”/”single”: which Workbench upload type we’re using.
- headers: the name of the headers file (no extension!), which is stored in “_headers”. This lists all the headers to fill. You should construct a file for your department of the headers you would like to use, making sure it includes all in _requiredBook or _requiredSingle.
- ArchivesSpace file, using --AS flag (optional): name of the ArchivesSpace spreadsheet to use

After running the script, you will get an Excel file in “metadata”. Open it and fill it out.

Validate_Filled.py

Checks a few fields in a filled “Fillable” Excel spreadsheet to confirm that some key fields are valid. The exact fields checked are printed out to the terminal.

Preparation for use:

- Put the filled Excel spreadsheet in “metadata”

Run the command line. Example: python Validate_Filled.py fillable_cnair_book.xlsx

Command line arguments:

- Filled Excel file: name of the file to use

This does not output any files, just displays the errors that have been caught.

Filled_to_WB.py

Takes a filled “Fillable” Excel spreadsheet (from Create_Fillable.py) and converts it to a Workbench csv for upload.

Preparation for use:

- Put the filled Excel spreadsheet in “metadata”

Run the command line. Example: python Filled_to_WB.py book fillable_cnair_book.xlsx

Command line arguments:

- “book”/”single”: which Workbench upload type we’re using
- Filled Excel file: name of the file to use

After running the script, you will get a Workbench csv file in “metadata”. Open it up into Excel to confirm it looks okay. You can make edits you want to this csv. See field explanations later in this doc for where this might make sense. Send the CSV and all the files (previously placed in “files_to_upload”) to the Workbench server using WinSCP.

Nodes_from_WB.py

Takes the nodes CSV that Workbench outputs, and creates an Excel file containing the fields that are useful for ArchivesSpace’s Digital Object creation.

Preparation for use:

- Put the nodes file (a .csv) into “metadata”

Run the command line. Example: python Nodes_from_WB.py nodes.csv

Command line arguments:

- Nodes file: name of the nodes csv file

After running the script, you will get an Excel file in “metadata”. You can use this a couple of ways:

- If you used an ArchivesSpace Bulk Update Spreadsheet in Create_Fillable.py, and none of the records have since been edited in ArchivesSpace, you can copy-paste this (starting with “digital_object_id”) straight into the Digital Object fields of the existing ArchivesSpace spreadsheet. The order, and field order, should exactly match, and you can confirm this by looking at “id” (Workbench’s id) and “title”. This can be run as a spreadsheet bulk update task in ArchivesSpace.
- If you did edit records, because the versions won’t match, ArchivesSpace would throw an error. But you can re-export the Bulk Update Spreadsheet and pop these in the same.
- You can use this to fill out fields manually in ArchivesSpace’s PUI, starting with “digital_object_id”.

File usage - ArchivesSpace record creation from audio

Given the relative rarity of this task, automating the generation of ArchivesSpace’s bulk update template seems unnecessary.

1. Workbench with ID3 in middle
    a. Run Create_Fillable.py from files
    b. Fill, then run Fillable_to_WB.py
    c. Create ID3 tag dictionary from Fillable and run against files using mp3tag (script for generation?). This applies tags to all the audio.
    d. Run Workbench
    e. Receive nodes from Workbench, run Nodes_from_WB.py

2. Create ArchivesSpace bulk ingest
    a. Copy-paste from Fillable into ArchivesSpace bulk ingest template
    b. Add Digital Object nodes to same template
    c. Save, add carrier records and change hierarchies so they sit correctly (script for this?)
    d. Run ArchivesSpace bulk ingest

How to fill a “Fillable” spreadsheet

See the field/”header” explanations separately.

Most map exactly to one or more Workbench fields. Very few fields that Workbench requires to function are left in here. Around half the fields will get filled in from file metadata or ArchivesSpace.

A few fields will fill downwards through blanks, so you don’t need to copy-paste. For example, if the whole Workbench sheet goes to one collection or one node, you can fill that in just the first time, and it will fill every blank afterwards. It will stop filling when it gets to another value.

After completing the Workbench sheet, you may want to make some other edits. Notably:

- There may be some other fields that were not part of the headers list, but are useful for this specific case. See note on “adding fields” for permanently adding fields.
- A language that does not have an ISO639 code should be added as “*Language name* (mis)”.
- This assumes you want to add to existing nodes. If you need to create collection nodes on the fly, fill coll_node in with an arbitrary number (e.g. 1), then add parent rows in the output, and fix the parent_id to id matching manually. There are too many potential bugs otherwise.

agents_from_title_and_refid.py

This is a short script that uses some of the same files to generate a .csv file of agents associated with specific ArchivesSpace Archival Objects. This is useful for retitling where agents need to appear in the title.

Maintenance

_CVs:

- agents_in_AS.csv: Run ArchivesSpace background job “Agents-Archival Objects”, which generates one row per Agent-Archival Object match. Reduce this file (in Libreoffice, as Excel can’t open a csv of this size) to get a csv of just agent, title, refid.
- cnair_culture.csv: copy-paste the list here: https://indigenousguide.amphilsoc.org/culture_browse/all
- iso639.csv: Run the file _ISO639json_to_CSV.py with argument as the filepath for the json from Lbeaudoux’s Github: https://github.com/LBeaudoux/iso639/blob/bb15c9fc5abfd094b3ebedee3c8f9a2bc628cfb9/iso639/data/iso-639.json

Adding fields:

- _headers contains files listing fields used by this process

- A user's set of used fields can be added to /_headers as a new csv with no underscore. Note that this should include all fields in _requiredBook or _requiredSingle or an error would be triggered. E.g. for CNAIR uploads, all CNAIR used fields are in csv files prefixed by CNAIR.
- Any field to be added in its entirety should first be added to files in _headers (_allValid, user sets, and _required* if required), and then added following the existing examples to Filled_to_WB.py and (if prefilling) to Create_Fillable.py