lFIscripts Documentation

Release 2018-08-09

Kieran O'Leary

Mar 15, 2024

CONTENTS:

CHAPTER

ONE

SUMMARY

These scripts facilitate collections management workflows within the IFI Irish Film Archive. These scripts have been tested in OSX & Mac OS, Windows 7 & 10 & 11, Ubuntu 14.04 & 16.04 & 18.04. They are mostly Python 3.7 compatible but some are still Python 2.7 only.

They are located here on github: https://github.com/Irish-Film-Institute/IFIscripts

An installation package version is available on PyPI: https://pypi.org/project/ifiscripts/. The scripts included can be found in `setup.py` or `pyproject.toml`. To install: run `pip(3) install ifiscripts` in the terminal.

Most scripts have ArgumentParser so you can run *$ifiscript.py -h* to check the usage. Some scripts without ArgumentParser take either a file or a directory as their input, for example `makeffv1.py filename.mov` or `premis.py path/to/folder_of_stuff`. For all the arguments requiring a path, it's best to just drag and drop the folder or filename into the terminal as this provides the absolute path.

We want the project to be as reuseable as possible in different institutions and contexts. Some scripts, particularly anything to do with `Object Entry` or `Accessioning` will be quite IFI specific, but other scripts such as `makeffv1.py`, `dcpaccess.py` and many others have been used in a variety of contexts in several different countries.

The project uses the MIT license, and we encourage the reuse, modification and study of the scripts. It's always nice to hear when the scripts have been reused in some way, but it's not necessary to let us know.

CHAPTER

TWO

PURPOSE

These python scripts facilitate much of our collections management procedures for digitised and born digital objects in the Irish Film Institute. We utilise a lot of open source tools, so we wanted to make these scripts as open as possible. This is why this project has the MIT License.

The Irish Film Institute has followed the SPECTRUM museum collections management standard for several years. These scripts attempt to follow SPECTRUM procedures while also utilising some of the concepts of the Open Archival Information System (OAIS). Initially the scripts only handled single video files, but they are now capable of handling:

- Digital Cinema Packages
- XDCAM cards
- DPX/TIFF image sequences
- Documents (.doc, .pdf etc)
- Images (.jpg, .TIFF etc)

An example workflow might be:

- A digital object is created or acquired by the IFI, and `ingest` begins.
- **`sipcreator.py` is run on the object. This script:**
  - generates an `Object Entry` identifier (eg OE-1234)
  - creates a folder structure for `logs, metadata, objects`
  - generates a `UUID`, extracts technical metadata
  - generates a md5 checksum manifest
  - and see the usage section for more.
- All of these preservation events are logged in a log file located in the `logs` directory. This log file tries to use `PREMIS (PREservation Metadata Implementation Strategies)` terminology as much as possible.
- Even though the package has yet to be accessioned, temporary backups are required. `copyit.py` will generate backups, and it will use the checksum manifest generated by `sipcreator.py` to verify the integrity of the file transfer.
- If the package contains FFV1 or Matroska files, perhaps `ffv1mkvvalidate.py` could run, which uses `mediaconch` to verify the compliance of the files, and stores the information in the logfile.
- If the package passes our Quality Control Procedures, then it will be accessioned. `aipcreator.py` will generate an accession number, rename the OE number with the accession number, generate a SHA-512 manifest and update the log file to document these new preservation events.
- A large batch of items can be AIPed using `batchaipcreator.py`. If the `-pbcore` command line argument is used with the aipcreator scripts, technical metadata based on the PBCore standard will be generated in CSV

format. This process can be run seperately by using `makepbcore.py`. CSV was chosen instead of XML as this allows us to immediately import the CSV into our database system so that we have item level records.

- Access copies may be needed, so low-res watermarked proxies can be generated with `bitc.py`, or high res mezzanines with `prores.py`.
- The AIP can then be written to preservation storage, again using the `copyit.py` command.

So this is just one way of using the scripts from acquisition to preservation storage, but there are many other scripts for specific workflows, which you can investigate further down in the documentation.

CHAPTER

THREE

TABLE OF CONTENTS

3.1 Installation

3.1.1 General

This is a python 3.8 project.

In general, you can install the scripts from Pypi (installed along with Python) by running `python -m pip install ifiscripts` on Windows or `python3 -m pip install ifiscripts` on Mac OS.

You can also clone or download the whole repository (https://github.com/Irish-Film-Institute/IFIscripts) and run the scripts from your cloned path.

For our developement workflows, on Mac OS and Windows, we create a folder in the home directory called `ifigit`, then we run `git clone https://github.com/Irish-Film-Institute/IFIscripts.git`. Then we add the `ifiscripts` folder to `$PATH` which allows us to access the scripts from any directory, not just from `ifigit/ifiscripts`.

Alternatively, some folks prefer to locate (`cd`) into the cloned repository and run the scripts from there, for example to run `makeffv1.py` `python makeffv1.py path/to.filename.mov`.

3.1.2 External Dependencies

External dependencies are listed below.

Some vital dependencies are for the library or the common tools used for the materials. They can be found in the `pyproject.toml` dependencies list and will be installed when installing ifiscripts using `pip`. If you are cloning the repository, you can manually install the dependencies, such as `pip install lxml`.

- bagit
- clairmeta
- dicttoxml
- future
- lxml
- psutil

Some essential `subprocess` dependencies not in Pypi for most of the scripts will have to be installed manually.

- ffmpeg
- ffprobe

- mediainfo

The following dependencies are also needed for many scripts:

- 7zip aka 7za aka p7zip-full
- asdcplib (https://github.com/cinecert/asdcplib, required by clairmeta for checking DCP)
- exiftool (https://exiftool.org/install.html)
- md5deep
- mediaconch
- qcli
- rawcooked
- rsync
- siegfried aka sf (https://www.itforarchivists.com/siegfried)
- sox (SoundeXchange, https://sourceforge.net/projects/sox/, required by clairmeta)

Some dependencies below are not our common dependencies but could be needed for specific purpose:

- gcp (copy from/to LTO, installed via gnu-coreutils on OSX)
- git (for update.py, no longer in use for updating scripts)
- mkvpropedit (for concat.py, installed via mkvtoolnix)
- openssl

3.1.3 Specific Instructions - Windows

- install 64-bit git-bash using all the default settings https://git-scm.com/downloads - make sure it's the 64-bit version!
- install 64-bit python3, making sure to tick the option to ADD TO PATH https://www.python.org/downloads/
- open cmd and `mkdir ifiscripts` and `git clone https://github.com/Irish-Film-Institute/IFIscripts.git`
- add this ifiscripts path (eg `C:\Users\$USER\ifigit\ifiscripts`)to the environmental path, following these steps: https://www.computerhope.com/issues/ch000549.htm
- ffmpeg the default option works well - 64-bit static https://ffmpeg.zeranoe.com/builds/ and place in scripts folder * OR install media-autobuild-suite but extract to the C:mas folder due to long path issues
- mediainfo - get the 64-bit CLI version https://mediaarea.net/en/MediaInfo/Download/Windows
- install lxml with `pip install lxml` if not installing ifiscripts by pip
- install siegfried exe (https://www.itforarchivists.com/siegfried/)file to the ifiscripts folder and run `sf -update` in cmd
- download exiftool installer and select the 'latest build' option - make sure that the option to add to path is ticked - https://oliverbetz.de/pages/Artikel/ExifTool-for-Windows
- install notepad++ - https://notepad-plus-plus.org/downloads/
- install libreoffice for accessing and editing csv files - https://www.libreoffice.org/download/download/

3.1.4 Specific Instructions - Ubuntu

(We no longer support Ubuntu but the scripts are still working)

A lot of these can be installed on Ubuntu with a single line: `sudo apt update && sudo apt install python3-pip ffmpeg mkvtoolnix exiftool git md5deep p7zip-full`

In order to add the rest, refer to the installation instructions of the relevant tools. For mediaarea tools, it can be easiest to use their own snapshot repository:

```
wget https://mediaarea.net/repo/deb/repo-mediaarea-snapshots_1.0-13_all.deb && sudo
dpkg -i repo-mediaarea-snapshots_1.0-13_all.deb && sudo apt update && sudo apt install
mediainfo dvrescue qcli rawcooked mediaconch
```

Please find an extra installation and update guideline for installing ifiscripts using *pip* on Windows or Mac. This instruction starts from installing python and includes basic details.

3.2 Contributing

3.2.1 Issues

Contributions are very much welcome in any form. Feel free to raise an issue requesting a new feature, or to report a bug. If requesting a new feature, please describe the workflow which the feature will be used, the required input & output data, and the output form of that. If reporting a bug, please copy/paste the full, complete, uncut terminal output.

For the issues requested by IFI Irish Film Archive staff, please raise tickets in the Teams/Script work group/Script_maintenance_log.xlsx. All tickets will be assessed with workflow first and then uploaded on GitHub in new branches or issues if needed.

For issues we are going to solve, we will create a relative pull request and quote each other.

3.2.2 Pull Requests

Pull requests are welcome. If contribution is on existing scripts, please follow the coding style and leave comments for short descriptions. If contributing new script, it can be nice to run it through `pylint` first, as this will check for PEP-08 compliance.

We are trying to limit the use of external dependencies unless it is necessary for the workflows and materials. While if you add any external dependencies, please specify in the pull request and add them in `setup.py`.

Please commit as '$script.py - $description' for a clear review. Please also leave the test instances and results with the full, complete, uncut terminal output. If it generates any files, please specify the filename and format, and content if applicable.

For pull requests by IFI Irish Film Archive staff, please follow the script work group policy and contributing after testing in the local workspace. For any edits of re-phrasing or typo, force-push would be better in case of unnecessary commits.

3.2.3 Contributions Needed

Some good first contributions could be adding explanatory docstrings to libraries like `ififuncs.py`, or revising older scripts, such as `validate.py` so that they are more in line with the coding style of recent scripts. Some of our `main()` functions are far too long and are doing too much, so they are in need of being split up into smaller functions.

Overall, the project needs to grow in the following ways:

- Reduce code duplication across scripts. These duplications continue to be difficult to maintain. Moving regularly used functions to ififuncs definitely helps.
- `unittests` are desperately needed! Scripts are becoming more and more linked, so automated testing is needed in order to find errors in areas that we might not expect. This should also allow integration with something like `Travis`.
- A config file is needed in some way shape or form. For example, logs and old manifests are stored on the desktop. It isn't really cool that these folders just appear without the user even knowing. This could also allow the scripts to be more generic, as IFI specific options could be enabled here.
- It would probably be a good idea to introduce classes into IFIscripts. Some functions return and call an embarrasing number of values.
- Have some sort of integration with a mysql database for tracking objects and logging events.

3.3 Usage

3.3.1 Arrangement

3.3.1.1 sipcreator.py

- Accepts one or more files or directories as input and wraps them up in a directory structure in line with IFI procedures using `copyit.py`.
- Source objects will be stored in an /objects directory. Directory structure is: parent directory named with a UUID, with three child directories (objects, logs metadata):
- Metadata is extracted for the AV material and MD5 checksums are stored for the entire package. A log records the major events in the process.
- Usage for one directory - `sipcreator.py -i /path/to/directory_name -o /path/to/output_folder`
- Usage for more than one directory - `sipcreator.py -i /path/to/directory_name1 /path/to/directory_name2 -o /path/to/output_folder`
- Run `sipcreator.py -h` for all options.

3.3.1.2 batchsipcreator.py

- Batch process packages by running `sipcreator.py`
- The script will only process files within subfolders.
- The script will ask for the starting OE number, and each further package will auto-increment by one.
- Usage for processing all subdirectories that (for example) places all XML/PDF/TXT files in the supplemental metadata subdirectory, and place all MF and STL files within objects- `batchsipcreator.py -i /path/to/directory_name -o /path/to/output_folder -supplement_extension_pattern xml pdf txt -object_extension_pattern mxf stl`
- Run `batchsipcreator.py -h` for all options.

3.3.1.3 aipcreator.py

- Turns a SIP that has passed QC procedures into an AIP.
- Currently this just works with packages that have been generated using `sipcreator.py` and `seq2ffv1.py`. SHA512 manifests are created,the OE number is replaced by an accession number, and the sipcreator logfile is updated with the various events that have taken place.
- Usage for one directory - `aipcreator.py /path/to/directory_name`
- Run `aipcreator.py -h` for all options.

3.3.1.4 batchaipcreator.py

- Batch process packages by running `aipcreator.py` and `makepbcore.py`
- The script will only process files with `sipcreator.py` style packages. `makeffv1.py` and `dvsip.py` packages will be ignored.
- Usage for processing all subdirectories - `batchaipcreator.py /path/to/directory_name`
- Run `batchaipcreator.py -h` for all options.

3.3.1.5 order.py

- Audits logfiles to determine the parent of a derivative package.
- This script can aid in automating large accessioning procedures that involve the accessioning of derivatives along with masters, eg a Camera Card and a concatenated derivative, or a master file and a mezzanine.
- Currently, this script will return a value :`None`, or the parent `OE` number. It also prints the OE number in its `OE-XXXX` just for fun.
- Usage for one directory - `order.py /path/to/directory_name`

3.3.1.6 makepbcore.py

- Describes AV objects using a combination of the PBCore 2 metadata standard and the IFI technical database.
- This script takes a folder as input. Either a single file or multiple objects will be described.
- This will produce a single PBCore CSV record per package, even if multiple objects are within a package. The use case here is complex packages such as XDCAM/DCP, where we want a single metadata record for a multi-file object.
- The CSV headings are written in such a way to allow for direct import into our SQL database.
- Usage for one directory - `makepbcore.py /path/to/directory_name`
- Run `makepbcore.py -h` for all options.

3.3.1.7 mergepbcore.py

- Collates PBCore CSV records into a single merged CSV.
- The merged csv will be stored in the Desktop ifiscripts_logs folder.
- This script takes a parent folder containing AIPs as input.
- Usage `mergepbcore.py /path/to/folder_that_contains_AIPs_as_input`
- Run `mergepbcore.py -h` for all options.

3.3.1.8 merge_csv.py

- Collates CSV records into a single merged CSV.
- The merged csv will be stored in the Desktop ifiscripts_logs folder. There is no error checking.
- This script takes as many as CSV files with the same titles as needed as input.
- Usage `merge_csv.py /path/to/csv_1 /path/to/csv_2 /path/to/csv_x`
- Run `merge_csv.py -h` for all options.

3.3.1.9 deletefiles.py

- Deletes files after `sipcreator.py` has been run, but before `aipcreator.py` has been run.
- Manifests are updated, metadata is deleted and the events are all logged in the logfile.
- This script takes the parent OE folder as input. Use the `-i` argument to supply the various files that should be deleted from the package.
- Usage for deleting two example files - `deletefiles.py /path/to/oe_folder -i path/to/file1.mov path/to/file2.mov`
- Run `deletefiles.py -h` for all options.

3.3.1.10 package_update.py

- Rearranges files into a subfolder files after `sipcreator.py` has been run, but before `aipcreator.py` has been run.
- Manifests are updated, files are moved, and the events are all logged in the logfile.
- This is useful in conjunction with `sipcreator.py` and `deletefiles.py`, in case a user wishes to impose a different ordering of the files within a large package. For example, from a folder with 1000 photographs, you may wish to create some sufolders to reflect different series/subseries within this collection. This script will track all these arrangement decisions.
- This script takes the parent OE folder as input. Use the `-i` argument to supply the various files that should be moved. The `new_folder` argument declares which folder the files should be moved into. Run `validate.py` to verify that all went well.
- Usage for moving a single file into a subfolder - `package_update.py /path/to/oe_folder -i path/to/uuid/objects/file1.mov -new_folder path/to/uuid/objects/new_foldername`
- Run `package_update.py -h` for all options.

3.3.1.11 subfolders.py

- Generates subfolders based on filenames within the input directory and if `-move` is used, moves the relevant files into these new directories.
- Eg. An input directory contains file1.mkv, file1.xml file2.mkv, file2.xml This will result in directories called file1 and file2 being created, and file1.mkv and file1.xml will be moved into the file1 directory, with a similar action for file2
- Usage to just make subfolders: `subfolders.py -i path/to/input`
- Usage to make subfolders and move files: `subfolders.py -move -i path/to/input`

3.3.1.12 accession_register.py

- Generates a helper accession register based on the metadata in other spreadsheets.
- Usage: `accession_register.py -pbcore_csv /path/to/pbcore_csv -sorted_csv /path/to/sorted_csv -filmo_csv /path/to/filmo_csv`

3.3.2 Transcodes

3.3.2.1 normalise.py

- Transcodes to FFV1/Matroska and performs framemd5 validation. Accepts single files only. Batch functionality may be added at a later date. For IFI purposes, the `-sip` option is needed as this will also launch `sipcreator.py` and generate the IFI package structure. If this `-sip` flag is not used, then the script will not impose a folder structure. You may wish to add some supplemtal metadata to the package, such as an EDL or some capture notes, so these can be added with the `-supplement` option.
- Currently, the lossless report is displayed in the middle of the process, so care is needed to ensure that the losslessness is verified before moving on to accessioning.
- Usage within IFI - `normalise.py -i filename.mov -o /path/to/output_directory -sip`

- Usage within IFI with supplement option - `normalise.py -i filename.mov -o /path/to/output_directory -sip -supplement path/to/supplemental_1.txt path/to/supplemental2.edl`
- Usage for single file in a general usage - `normalise.py -i filename.mov -o /path/to/output_directory`

3.3.2.2 makeffv1.py

- Transcodes to FFV1.mkv and performs framemd5 validation. Accepts single files or directories (all video files in a directory will be processed). CSV report is generated which gives details on losslessness and compression ratio.
- Usage for single file - `makeffv1.py filename.mov`
- Usage for batch processing all videos in a directory - `makeffv1.py directory_name`

3.3.2.3 bitc.py

- Create timecoded/watermarked h264s for single files or a batch process.
- Usage for single file - `bitc.py filename.mov`
- Usage for batch processing all videos in a directory - `bitc.py directory_name`
- This script has many extra options, such as deinterlacing, quality settings, rescaling. Use `bitc.py -h` to see all options

3.3.2.4 prores.py

- Transcode to prores.mov for single/multiple files.
- Usage for single file - `prores.py filename.mov`
- Usage for batch processing all videos in a directory - `prores.py directory_name`
- This script has many extra options, such as deinterlacing, quality settings, rescaling. Use `prores.py -h` to see all options

3.3.2.5 makedip.py

- Runs bitc.py or prores.py.
- Usage for running bitc.py on all objects in a batch of information packages - `makedip.py path/to/batch_directories -o path/to/output`
- The `-prores` option will use run `prores.py` instead of `bitc.py`
- The script will rename the output file so that it contains either the OE number or the accession number.
- If it sees that a proxy already exists, then it will skip the video.
- Use `makedip.py -h` to see all options

3.3.2.6 concat.py

- Concatenate/join video files together using ffmpeg stream copy into a single Matroska container. Each source clip will have its own chapter marker. As the streams are copied, the speed is quite fast.
- Usage: `concat.py -i /path/to/filename1.mov /path/to/filename2.mov -o /path/to/destination_folder`
- A lossless verification process will also run, which takes stream level checksums of all streams and compares the values. This is not very reliable at the moment.
- Warning - video files must have the same technical attributes such as codec, width, height, fps. Some characters in filenames will cause the script to fail. Some of these include quotes. The script will ask the user if quotes should be renamed with underscores. Also, a temporary concatenation textfile will be stored in your temp folder. Currently only tested on Ubuntu.
- Dependencies: mkvpropedit, ffmpeg. ## Digital Cinema Package Scripts ##

3.3.2.7 dcpaccess.py

- Create h264 (default) or prores transcodes (with optional subtitles) for unencrypted, single/multi reel Interop/SMPTE DCPs. The script will search for all DCPs in subdirectories, process them one at a time and export files to your Desktop.
- Usage: `dcpaccess.py dcp_directory`
- Use `-p` for prores output, and use `-hd` to rescale to 1920:1080 while maintaining the aspect ratio.
- Dependencies: ffmpeg must be compiled with libopenjpeg - `brew install ffmpeg --with-openjpeg`.
- Python dependencies: lxml required.
- Further options can be viewed with `dcpaccess.py -h`

3.3.2.8 dcpfixity.py

- Verify internal hashes in a DCP and write report to CSV. Optional (experimental) bagging if hashes validate. The script will search for all DCPs in subdirectories, process them one at a time and generate a CSV report.
- Usage: `dcpfixity.py dcp_directory`
- Further options can be viewed with `dcpfixity.py -h`

3.3.2.9 dcpsubs2srt.py

- Super basic but functional DCP XML subtitle to SRT conversion. This code is also contained in dcpaccess.py
- Usage: `dcpsubs2srt.py subs.xml`

3.3.3 Fixity Scripts

3.3.3.1 copyit.py

- Copies a file or directory, creating a md5 manifest at source and destination and comparing the two. Skips hidden files and directories.
- Usage: `copyit.py source_dir destination_dir`
- Dependencies: OSX requires gcp - `brew install coreutils`

3.3.3.2 manifest.py

- Creates relative md5 or sha512 checksum manifest of a directory.
- Usage: `manifest.py directory` or for sha512 hashes: `manifest.py -sha512 directory`
- By default, these hashes are stored in a desktop directory, but use the `-s` option in order to generate a sidecar in the same directory as your source.
- Run `manifest.py -h` to see all options.

3.3.3.3 makedfxml.py

- WARNING - until this issue is resolved, this script can not work with Windows: https://github.com/simsong/dfxml/issues/29
- Prints Digital Forensics XML to your terminal. Hashes are turned off for now as these will usually already exist in a manifest. The main purpose of this script is to preserve file system metadata such as date created/date modified/date accessed.
- This is a launcher script for an edited version of 'https://github.com/simsong/dfxml/blob/master/python/walk_to_dfxml.py'. The edited version of `walk_to_dfxml.py` and the `Objects.py` library have been copied into this repository for the sake of convenience.
- Usage: `makedfxml.py directory`.
- NOTE: This is currently a proof of concept. Further options, logging and integration into other scripts will be needed.
- There may be a python3 related error on OSX if python is installed via homebrew. This can be fixed by typing `unset PYTHONPATH` in the terminal.

3.3.3.4 shadfxml.py

- Creates DFXML and sha512 manifests but only in sipcreator/uuid packages.
- This will work recursively so all packages within a directory will be processed.
- Usage: `shadfxml.py directory`

3.3.3.5 validate.py

- Validate md5 or SHA512 sidecar manifests. Currently the script expects two spaces between the checksum and the filename.
- In packages that have been generated with sipcreator.py, the results of the process will be added to the logfile and the checksum for the logfile will update within the md5 and sha512 manifests
- Usage: `validate.py /path/to/manifest.md5` or `validate.py /path/to/_manifest-sha512.txt`

3.3.3.6 as11fixity.py

- Validates AS-11 UK DPP mxf file(s) by comparing file checksum from both information package manifest and DPP xml <MediaChecksumValue>.
- Prints file & folder count and fixity details for each package.
- Generates a csv report on the desktop for checking details.
- Usage: `as11fixity.py /path/to/as_11` or `as11fixity.py /path/to/parent_folder`
- `as11fixity.py -h` is also available

3.3.3.7 batchdiff_framemd5.py

- Creates framemd5 sidecars on a batch of SIPs powered by *framemd5.py*; Compares the hashes in framesmd5 and those in md5 files in PSM directory; Once mismatch was found, it will skip the rest of the hashes and skip to the next object; It will delete all framemd5 files after the batch of the comparsions have finished.
- Usage: `batchdiff_framemd5.py -sip /path/to/parent_folder/of/SIPs -psm /path/to/parent_folder/of/PSMs`
- NB: The script will default to only one md5 manifest file per PSM. If there are repeated manifest in the directory, users may need to add bloack in the script manually.

3.3.4 Image Sequences

3.3.4.1 seq2ffv1.py

- Work in progress -more testing to be done.
- Recursively batch process image sequence folders and transcode to a single ffv1.mkv.
- Framemd5 files are generated and validated for losslessness.
- Whole file manifests are also created.
- Usage - `seq2ffv1.py parent_folder`

3.3.4.2 seq2prores.py

- Specific IFI workflow that expects a particular folder path:
- Recursively batch process image sequence folders with seperate WAV files and transcode to a single Apple Pro Res HQ file in a MOV container. PREMIS XML log files are generated with hardcoded IFI values for the source DPX sequence and the transcoded mezzanine file in the respective /metadata directory
- A whole file MD5 manifest of everything in the SIP are also created. Work in progress - more testing to be done.
- Usage - `seq2prores.py directory`
- seq2prores accepts multiple parent folders, so one can run `seq2prores.py directory1 directory2 directory3` etc

3.3.4.3 seq.py

- Transcodes a TIFF sequence to 24fps v210 in a MOV container.
- Usage: `seq.py path/to/tiff_folder` and output will be stored in the parent directory.
- Further options can be viewed using `seq.py -h`

3.3.4.4 oeremove.py

- IFI specific script that removes OE### numbers from the head of an image sequence filename.
- Usage - `oeremove.py directory`.

3.3.4.5 seq2dv.py

- Transcodes a TIFF sequence to 24fps 720x576 DV in a MOV container.
- Usage: `seq.py path/to/tiff_folder` and output will be stored in the parent directory.

3.3.4.6 batchmetadata.py

- Traverses through subdirectories trying to find DPX and TIFF files and creates mediainfo and mediatrace XML files.
- Usage: `batchmetadata.py path/to/parent_directory` and output will be stored in the parent directory.

3.3.4.7 batchrename.py

- Renames TIFF files in an image sequence except for numberic sequence and file extension.
- Usage - `batchrename.py directory` - enter new filename when prompted

3.3.5 Quality Control

3.3.5.1 massqc.py

- Generate QCTools xml.gz sidecar files via `qcli` which will load immediately in QCTools.
- Usage for single file - `massqc.py filename.mov`
- Usage for batch processing all videos in a directory - `massqc.py directory_name`

3.3.5.2 videoerror.py

- Detect corrupted frames in m2t/HDV captures.
- Generates a CSV report in ~/Desktop/ifiscripts_logs
- Usage for batch processing all m2t videos recursively in a directory - `` videoerror.py directory_name``

3.3.5.3 framemd5.py

- Creates framemd5 sidecars on all mov/mkv files in all subfolders beneath your input.
- If the input is a file, then `framemd5.py` will just generate a sidecar for this one file.
- Usage for single file - `framemd5.py -i filename.mov`
- Usage for batch processing all videos in a directory - `framemd5.py -i directory_name`

3.3.5.4 ffv1mkvvalidate.py

- Validates Matroska files using mediaconch.
- An XML report will be written to the metadata directory.
- A log will appear on the desktop, which will be merged into the SIP log in /logs.
- Usage for batch processing all videos in a directory - `ffv1mkvvalidate.py directory_name`

3.3.5.5 lossy_check.py

- This script is to check losslessness for a batch of sipped image sequence objects
- It will check the losslessness from package/$uuid/logs/$uuid_seq2ffv1_log.log
- It will return the result of ‘lossless’ or ‘lossy’ for each information package
- Usage for batch processing all videos in a directory - `lossy_check.py -i directory_name`

3.3.5.6 structure_check.py

- This script is to check the structure of a batch of SIPs/AIPs(and AIP shells)
- It will show the directory tree of each information package
- Users are able to manually record if the structure is right or not
- The script will list a summary at the end
- Usage: `structure_check.py -i directory_name`

3.3.6 Specific Workflows

3.3.6.1 masscopy.py

- Copies all directories in your input location using copyit.py ONLY if a manifest sidecar already exists.
- This is useful if a lot of SIPs produced by makeffv1 are created and you want to move them all to another location while harnessing the pre-existing checksum manifest.
- WARNING - It is essential to check the log file on the desktop/ifiscripts_logs for each folder that transferred!!
- Usage: `masscopy.py /path/to/parent_folder -o /path/to/destination_folder`

3.3.6.2 makefolders.py

- Creates a logs/objects/metadata folder structure with a UUID parent folder. This is specific to a film scanning workflow as there are seperate audio and image subfolders. You can specifiy the values on the command line or a terminal interview will appear which will prompt you for filmographic URN, source accession number and title. Use `makefolders.py -h` for the full list of options.
- Usage: `makefolders.py -o /path/to/destination`

3.3.6.3 loopline_repackage.py

- Retrospectively updates older FFV1/DV packages in order to meet our current packaging requirements. This should allow aipcreator.py and makepbcore.py to run as expected. This will process a group of packages and each loop will result in the increment by one of the starting OE number. Use with caution.
- This script should work on files created by `makeffv1.py dvsip.py loopline.py`
- Usage: `loopline_repackage`

3.3.6.4 batchmakeshell.py

- Creates shells for the AIPs under a batch (or SIP shells of large-size materials for backup use). This is used for the accessioning closing steps. The script will recognise all the folders named with "aaa[0-9]{4}" digital accession number format. Then created their shell folders named "aaa[0-9]{4}_shell" and clone all the subcontent except the content inside the 'objects' folder into them. The shells will be created into the targeted output path.
- Usage: `batchmakeshell.py path/to/batch_directories -o /path/to/destination`
- This script has extra options, including making shells for AS-11 UK DPP and DCP, and making SIP shells for DCDM and other large-size materials. Use `batchmakeshell.py -h` to see all options.

3.3.6.5 getdip.py

- Retrieves DIPs (shells, proxies or mezzanines) from storage by accession numbers. The script checks if the required AIPs are in the storage for a form of DIPs and copied to destination if needed. The accession numbers can be either input manually or imported by a formatted csv including the required accession numbers.
- Usage: `getdip.py -t DIP_type -n accession_numbers -i /root/of/DIP/dir -o path/to/destination` for manually typing the accession numbers in and copy the DIPs. `getdip.py -t DIP_type -csv path/to/accession_numbers/csv -i /root/of/DIP/dir -justcheck` for importing access numbers from csv and just checking if the required accession numbers have related DIP in the storage.
- Use `getdip.py -h` to see all options.

3.3.6.6 Special Collections

batchsc_checkdir.py

- Checks if each special collections package in the storage is an AIP (processed package) or a RAW (unprocessed package) has been processed, and then move either to a separate folder.'
- This script doesn't have any arguments. It prints 2 functions for selection.
- Usage: `batchsc_checkdir.py`

batchsc_validate.py

- Runs `validate.py` for a batch of special collections unprocessed packages under the same directory. All validated packages will be moved to required destination.
- All packages failed validation remain in the source location. The script will return the directory for the running data for all failed validations after it finishes.
- For all the backlogs having logs and md5 manifests, the script will remove the logs and manifests after they passed validation and moved to the destination.
- This script doesn't have any arguments. It prints the steps the script will do and ask for the source and the destination at the beginning.
- Usage: `batchsc_validate.py`

batchsc_organise.py

- Use with caution! It will change the files matadata and package structure and there is no log for this.
- Function 1 - (Use with caution) Rename folders/files - replace all spaces and special characters to '_'
- Function 2 - Rename packages - replace spaces to '_' and remove all special characters
- Function 3 - Move subfiles - move all files to the root of the packages
- Function 4 - (Use only after 3.) Delete subfolders - Delete all subfolders in each packages
- This scripts doesn't have any arguments. It asks for the source and lists all the packages. Then it prints above 4 functions for selection.

batchsc_aip_update.py

- Moves all subfiles in the objects folder in the AIPs (processed packages) to the root, or renames (removes special characters) all the subfiles. It does update the log for AIPs when each AIP completes processing. It will returns AIPs cannot update the log at the end.
- Usage: `batchsc_aip_update.py -movetoobjects /path/to/parent/folder` or `batchsc_aip_update.py -rename /path/to/parent/folder`

3.3.7 Misc

3.3.7.1 update.py

- Updates IFIscripts to the latest git head if the following directory structure exists in the home directory: `ifigit/ifiscripts`
- Usage: `update.py`

3.3.7.2 makeuuid.py

- Prints a new UUID to the terminal via the UUID python module and the create_uuid() helper function within ififuncs.
- Usage: `makeuuid.py`

3.3.7.3 durationcheck.py

- Recursive search through subdirectories and provides total duration in minutes. Accepts multiple inputs but provides the total duration of all inputs.
- Usage: `durationcheck.py /path/to/parent_folder` or `durationcheck.py /path/to/parent_folder1 /path/to/parent_folder2 /path/to/parent_folder3`

3.3.7.4 fakexdcam.py

- Creates a fake XDCAM EX structure for testing purposes
- Usage: `fakexdcam.py /path/to/output_folder`

3.3.7.5 get_ps_list.py

- Create a csv file from mounted 'preservation storage' directory, including accession numbers from package title, object entry number from log file, and accession numbers from log file for all preserved AIPs.
- Usage: `get_ps_list.py -i /path/to/preservation_storage`

3.3.7.6 check_register.py

- Print mismatch accession numbers and object entry numbers.
- Compare between csv files from preservation list by *get_ps_list.py* and from (partly) digital accession register or help register, etc.
- Usage: `check_register.py -preservation_storage_csv /path/to/csv_made_by_get_ps_list_py -register_csv /path/to/accession_register`

3.3.8 Experimental-Premis

3.3.8.1 premis.py

- Work in progress PREMIS implementation. This PREMIS document will hopefully function as a growing log file as an asset makes its way through a workflow.
- Requries pyqt4 (GUI) and lxml (xml parsing)
- Usage - `premis.py filename`.

3.3.8.2 viruscheck.py

- Work in progress script by @ecodonohoe
- Scans directories recursively using ClamAV

3.4 Credits

The majority of the code in this repository is written by Kieran O'Leary, but it is the product of many years of research by the staff within the Irish Film Institute. The actual code contributors can be seen here.

Past and present staff members who have contributed in one way shape or form to this project are [in NO order]:

Anja Mahler, Raelene Casey, Kasandra O'Connell, Kieran O'Leary, Eoin O'Donohoe, Aoife Fitzmaurice, Brian Cash, Columb Gilna, Manus McManus, Gavin Martin, Felix Meehan, Fiona Rigney, Michael Ryan, Dean Kavanagh, Sunniva O'Flynn, Eilís Ní Raghallaigh, Joanne Carroll, Karen Wall, Rebecca Grant, Simon Factor, Aaron Healy.

The project is very much inspired by the work of Dave Rice, particularly the `makelossless` script within mediamicroservices.

Other key external people who have contributed (the scripts would not be as they are without these people) in one way shape or form, whether they realise it or not are in NO order :

Ashley Blewer, Piaras Hoban, Reto Kromer, Dave Rice, Jerome Martinez, Peter Bubestinger, Maureen Callaghan, Peggy Griesinger, Kara Van Malssen, Lauren Sorenson, Kate Murray, Misty de Meo, Carl Eugen Hoyos, Michael Niedermayer, Vittorio Giovara, Kieran Kunhya, Stephen McConnachie, Edward Anderson, Peter Ross, Nick Krabbenhoeft, Andy Jackson, David Underdown, Rebecca Grant, Sandra Collins, Ana Ribieiro, Patricia Falcao, Fergus O'Connor, Ben Fino-Radin, Michael Campos-Quinn, Mike Casey, Charles Hosale, Ben Turkus, Kelly Haydon, Micky Lindner, Alessandra Luciano, Jonáš Svatoš, Andrew Weaver, Libby Hopfauf, Bleakely McDowell, Siobhan C Hagan, Somaya Langley, Alexander Ivash, Moritz Barsnick, Moritz Bunkus, Mary Kidd, Elizabeth England, Zeranoe, Eddy Colloton, Ethan Gates, Savannah Campbell, Vicky Philips, Adam Lott, Yvonne Ng, Erwin Verbruggen, Carl Fleischauer, Lorena Ramirez-Lopez, Dinah Handel, Jim Linder, Tim Walsh, Johan van der Knijff, Donal Foreman, Jenny Hunt, John Warburton, Mark Philips, Steve Lhomme, Brendan Coates, Morgan Oscar Morel, Andreas Romeyke, Marion Jaks, Hermann Lewetz, Joshua Ng, Derek Buitenhuis, Tod Robbins, Katherine Frances Nagels, Jim Sam, Genevieve Havermayer-King, Jenny Mitcham, Brian tvc15, Zach Kelling, Ian Easton, Denis Warburton, Kathryn Gronsbell, Kathryn Cassidy,

Stuart Kenny, Richard Lehane, Paul B Mahol, Jonathan Farbowitz, Treasa Harkin, Ed Summers, Justin Simpson, oioi-iooixiii, Shira Peltzman, Dolores Grant, Bertram Lyons, The Great Bear, Big Bear, Se Merry Doyle, Eugene Finn, Jaime Mears, Richard Wright, Brecht Declercq, Nicole Martin, Rebecca Fraimow, John Resig, Casey Davis Kaufman, Charles Poynton, Michael Dineen, Mark Piggot, Emily Boylan, David O'Leary, Andrew Reid, John Gunn, Kai Man Wong, Lok Cheung.

Note: The original markdown documentation template was copied from mediamicroservices

NOTE: `Objects.py` has been copied from https://github.com/simsong/dfxml. `walk_to_dfxml.py` has also been copied but has been customised in order to add command line arguments for optionally turning off checksum generation. For more context, see https://github.com/simsong/dfxml/pull/28