# NAME

**DamCompare** – Daminion database verification

# SYNOPSIS

Python **DamScan.py** [-f] [-i]  
[-l] [-c *DBNAME*] [-s *SERVER*] [-p *PORT*] [-u *USER*]  
[-o *OUTFILE*]  
[-v] [-h] [--version]

# DESCRIPTION

Daminion digital asset management (DAM) system is a great tool for assigning meta data (tags) to your digital assets and for sorting and searching the items. Daminion writes all tags (with few exceptions) into the media files, so that the metadata is also available outside Daminion. In Daminion there is no option to verify if the metadata in the media items is the same as is in Daminion database. **DamCompare.py** solves this problem and reports inconsistencies in metadata for Daminion server and standalone catalogs.

The **DamScan.py** tool analyzes a Daminion catalog for potential inconsistencies in tagging between Daminion database and the media items. The analysis is done by comparing two Daminion databases. The first one is the current active/production catalog and the second one is current view of the image files. The current view is generated by creating an empty catalog and importing into it the same images that are in the first catalog. The import process will take time, because the speed can be as low as ~15 items/minute.

The program checks the values of these tags: *Creation Date/Time,* *Place, GPS, Event, People, Categories,* *Keywords* and *Collections*. The program also reports, if the media item is referring to a non-existing file. The command line options provide the capability to configure the process of the analysis as well as contents and format of its output.

For each media item the program reports the filename and the mismatching tag categories. It’s recommended to use **-i** option, so that it’s easy to find the mismatching items from the original and copy catalogs.

The output is tab separated so it can be directly pasted or (when using option **-o**) imported into a spreadsheet for further formatting, sorting and analyzing.

Options:

**-f, --fullpath** Display the full path and not only the file name

**-i, --id** Display the Daminion Item Id after the filename

**-l, --sqlite** Use a standalone (based on SQLite) catalog instead of server catalog (based on Postgresql)

**-c1, --catalog1** *CATALOG*The **-c1** option specifies the original Daminion catalog name. For standalone catalogs the full path and filename (including .dmc) must be specified.

**-c2, --catalog2** *CATALOG*The **-c2** option specifies the new created Daminion catalog against which the comparison is made. For standalone catalogs the full path and filename (including .dmc) must be specified.

**-s, --server** *SERVER*Postgres database server (**Not** the Daminion Server). If *SERVER* is not specified, localhost will be used. You can verify the *SERVER* and *PORT* settings in the Daminion Server Administration panel.

**-p --port** *PORT* Postgres database server port for the catalog. If not specified, the Daminion default *5432* will be used.

**-u --user** *USER*   
Postgres database user/password (**Not** Daminion catalog user). If not specified the installation default *postgres/postgres* will be used.

**-o --output** *OUTFILE*  
Write the report to an *OUTFILE*. If **-o** is not specified the output will be printed on the screen. Verbose messages (**-v**) are never directed to *OUTFILE.*

**-v, --verbose** Verbose output. Specifying the option **-v** displays a running counter, and the current Item Id and filename on the screen.

**-h, --help** Show help message and exit

**--version** Display version information and exit.

# DIAGNOSTICS

Errors and warnings are logged to the standard error stream and the diagnostic output to the standard output or the specified *OUTFILE*. If **-v** is not used, then no output means that no discrepancies were detected.

**DamCompare.py** terminates with zero exit status if it was able to scan through the whole catalog.

Only the specified tag categories (Creation Date/Time, Event, Place, GPS, People, Keywords, Categories, Collections) are checked, not all tag categories that are supported by Daminion. The other tag categories not in the list (e.g. Media Format, Rating, Project etc.) are ignored by **DamCompare.py**.

# ENVIRONMENT

## Python

Install Python 3.x from <http://www.python.org>. After you have downloaded Python package right click the package and select "*Run as administrator*". In the installation dialog select Customized installation. In the customized configuration tick to include Python in the PATH and select installation for all users. Other options can be left to defaults.

To activate the PATH settings, Windows should be restarted before starting Python for the first time.

## psycopg2

After installing Python start an elevated command window (*Run as Administrator*), because the Postgres support package will be installed in the Program Files directory. Enter commands

C:> python -m pip install -U pip setuptools

C:> python -m pip install psycopg2

# CONFIGURATION PARAMETERS

# SEE ALSO

[python](http://www.python.org), language description and syntax.

[psycopg2](https://pypi.python.org/pypi/psycopg2/), Python-PostgreSQL Database Adapter.

# LICENSE

The program is licensed under GPL3.

# AUTHOR(S)

Juha Lintula