NAME

DamScan – Daminion database analyzer

SYNOPSIS

```
Python DamScan.py [-f] [-i] [-g]
      [-t [{Event, Place, GPS, People, Keywords, Categories} [...]]]
      [-b [SEPARATOR [SEPARATOR ...]]]
      [-x EXFILE | -y ONLYFILE]
      [-l] [-c DBNAME] [-s SERVER] [-p PORT] [-u USER]
      [-o OUTFILE]
      [-v[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. In Daminion it's also possible to link or group associated items together, but there are no built-in tools for checking the consistency of the meta data of these linked or grouped items. **DamScan.py** solves this problem and reports inconsistencies in meta data for Daminion server and standalone catalogs.

The **DamScan.py** tool analyzes a Daminion catalog for potential inconsistencies in tagging between linked or grouped images. The program checks tag values in the tags: *Place, GPS, Event, People, Categories* and *Keywords*. It's also possible to check if the linked/grouped files have same base name. The command line options provide tools to configure the contents of the analysis.

For single value tags (*Place, GPS* and *Event*) and for the base filename the program reports both file names and both tag values (not for base names). For the multi value tags (*People, Categories* and *Keywords*) the program will report both filenames and which tag values **are missing** from the first file. The output is tab separated so it can be directly pasted or opened in a spreadsheet for further sorting and analyzing.

By default the program analyzes which images are linked together and compares the tag values between each linked image pair. When specifying **-g/--group** option the behavior is changed so that each image in the group is compared with the top image.

Options:

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

-i, --id Print database id after the filename

-g, --group Compare tags between grouped images instead of linked images, if this option is not specified, links are used

-t, --tags [{Event,Place,GPS,People,Keywords,Categories} [{Event,Place,GPS, ... }] ...]
Specify the tag categories that will be checked. If **-t** option is not specified all categories are checked. Allowed values for the list are *Event*, *Place*, *GPS*, *People*, *Keywords* and *Categories*. Multiple values are separated by space (‘ ‘). Specifying **-t** without any category disables the tag value checking. This can be useful with the option **-b** to check only the file names.

-b, --basename [*SEPARATOR* [*SEPARATOR* ...]]
Compare the base names of the files. If this option is not specified, only tag values are checked. If no *SEPARATOR* values are specified only the file extension is removed. If SEPARATOR values are specified, the file name is stripped until the last occurrence of *SEPARATOR*. If multiple values are specified each *SEPARATOR* is used once and the modified filename is checked for the next *SEPARATOR*. The same *SEPARATOR* value can be specified several times. If the base name would become shorter than 8 characters, the current and remaining *SEPARATOR*s will be ignored. **Note** the resulting base name for comparison can be different depending on the order of the *SEPARATOR*s.

-x, --exclude *EXFILE*
Configuration file for tag values that are excluded from comparison. The file format is like the standard INI file. See details in the section CONFIGURATION PARAMETERS. Either **-x** or **-y** parameter can be specified, not both. If neither option is specified all tag values are checked.

-y, --only *ONLYFILE*
Configuration file for the only tag values that are used for comparison. . The file format is like the standard INI file. See details in the section CONFIGURATION PARAMETERS. Either **-x** or **-y** parameter can be specified, not both. If neither option is specified all tag values are checked. Using the option **-y** with the same file as used with the option **-x**, it’s possible to verify what was/will be excluded.

-l, --sqlite Use standalone (Sqlite) catalog instead of server catalog (Postgresql)

-c, --catalog *CATALOG*
Daminion catalog name. If not specified the default Daminion Server catalog (*NetCatalog*) is used. 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 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*
: Print the report into *OUTFILE*. If **-o** is not specified the output will be printed on the screen. Verbose messages (**-v**) are not directed into *OUTFILE*.

-v, --verbose Verbose output. Specifying the option **-v** prints running counter, and the current database id and filename on the screen. If a second v is added to the option (**-vv**) also information of the linked/grouped pairs is printed. This output is always printed on the screen (stdout)

-h, --help Show help message and exit

--version Display version information and exit.

DIAGNOSTICS

Errors 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.

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

Using **-g** option requires time on larger catalogs, because the scanning is n^2 dependent on the number of items in catalog. As a benchmark for 150k items the analysis takes on SSD 1–2 hours and on HDD twice as long.

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.

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

The configuration parameters in the *EXFILE/ONLYFILE* follow the standard INI file structure. Each tag category can be specified in brackets ([category]) and the tag values below the section. If the tag value is hierarchical, the separator between hierarchy levels is '|'. If you specify only top items of a value hierarchy the filter applies to all child values.

Both the category names and tag values are case sensitive. You can specify also comment lines starting with '#' or ';'.

Below is an example configuration file.

```
[People]
# will be an exact match
Lintula|Juha

[Categories]
# will match to Image|B&W, Image|HDR, Image|Panorama, ...
Image

[Place]
# will match all cities and locations in Germany|Bavaria
Germany|Bavaria

[GPS]
# GPS coordinates not specified
0N 0E 0m
```

EXAMPLEs

Examples below assume that you have DamScan.py in your home directory (`C:\Users\user`) and your local catalog and the configuration files are in the `Pictures` sub-directory.

`C:> python DamScan.py -s ServerPC -p 5433`
Run the analysis of the NetCatalog server catalog, based on linked images. The Postgres database is set up in ServerPC at port #5433.

```
C:> python DamScan.py -v -g -l -c Pictures\DaminionCatalog.dmc -o
   Pictures\output.txt
```
Run the analysis of the local catalog `DaminionCatalog.dmc` in the `Pictures` directory, based on image groups. Print the results of the analysis in `Pictures\output.txt` and show basic progress information on the screen.

`C:> python DamScan.py -t Place GPS -c NewCatalog -p 5433`
Run the analysis only on Place and GPS tags of the NewCatalog server catalog, based on linked images. The Postgres database is set up in localhost at port #5433.

```
C:> python DamScan.py -vv -x Pictures\ExcludeList.ini -c NewCatalog -o
   Pictures\output.txt
```

Run the analysis on all tag categories of the NewCatalog server catalog, but exclude the defined tag values from comparison. Print the results of the analysis in `Pictures\output.txt` and show detailed progress information on the screen.

```
C:> python DamScan.py -y Pictures\ExcludeList.ini -c NewCatalog
```

Check what was excluded from the report of the previous example.

```
C:> python DamScan.py -b -t -l -g -c Pictures\TestCatalog.dmc
```

Analyze only the filenames of the local `TestCatalog.dmc`, based on grouped images. Only the file extension is removed before the comparison. I.e. `IMG_1234.CR2` will be the same as `IMG_1234.JPG`, but different from `IMG_1234_lowers.JPG`.

```
C:> python DamScan.py -b _ BW -t -l -g -c Pictures\TestCatalog.dmc
```

Same as the previous example, but more logic for comparing the file names. First the file extension is removed. Then everything until the last '_' and after that everything until last 'BW'. As a result all files `IMG_1234.JPG`, `IMG_1234_lowers.JPG`, `IMG_1234BW_lowers.JPG` and `IMG_1234BW.JPG` will match `IMG_1234.CR2`.

SEE ALSO

python, language description and syntax.

psycopg2, Python-PostgreSQL Database Adapter.

LICENSE

The program is licensed under GPL3.

AUTHOR(S)

Juha Lintula