datalad$_{m}etaladDocumentation$

DataLad team

May 08, 2019

Contents

The extension equips DataLad with some extra commands that enable alternative workflows.

CHAPTER 1

What is in it for users?

CHAPTER 2

What is in it for developers?

CHAPTER 3

API

3.1 High-level API commands

meta_extract([dataset, path, sources, ...])	Run one or more metadata extractors on a dataset or file.
meta_aggregate([path, dataset, recursive, ...])	Aggregate metadata of one or more (sub)datasets for later reporting.
meta_report([path, dataset, reporton, recursive])	Query a dataset's aggregated metadata for dataset and file metadata

3.1.1 datalad.api.meta_extract

datalad.api.**meta_extract**(*dataset=None*, *path=None*, *sources=None*, *process_type=None*, *format='native'*)

Run one or more metadata extractors on a dataset or file.

This command does not modify a dataset, but may initiate required data transfers to perform metadata extraction that requires local file content availability. This command does not support recursion into subdataset.

The result(s) are structured like the metadata DataLad would extract during metadata aggregation (in fact, this command is employed during aggregation). There is one result per dataset/file.

Examples

Extract metadata with two extractors from a dataset in the current directory and also from all its files:

```
$ datalad extract-metadata -d . --source xmp --source metalad_core
```

Extract XMP metadata from a single PDF that is not part of any dataset:

```
$ datalad extract-metadata --source xmp Downloads/freshfromtheweb.pdf
```

Customization of extraction:

The following configuration settings can be used to customize extractor behavior

datalad.metadata.extract-from-<extractorname> = {all|dataset|content} which type of information an enabled extractor will be operating on (see –process-type argument for details)

datalad.metadata.exclude-path = <path> ignore all content underneath the given path for metadata extraction, must be relative to the root of the dataset and in POSIX convention, and can be given multiple times

Parameters

- **dataset** (*Dataset or None, optional*) – "Dataset to extract metadata from. If no further constraining path is given, metadata is extracted from all files of the dataset. [Default: None]
- **path** (*sequence of str or None, optional*) – Path of a file to extract metadata from. [Default: None]
- **sources** – Name of a metadata extractor to be executed. If none is given, a set of default configured extractors, plus any extractors enabled in a dataset's configuration and invoked. Multiple extractors can be given as a list. [Default: None]
- **process_type** (*{None, 'all', 'dataset', 'content', 'extractors'}, optional*) – type of information to process. If 'all', metadata will be extracted for the entire dataset and its content. If not specified, the dataset's configuration will determine the selection, and will default to 'all'. Note that not processing content can influence the dataset metadata composition (e.g. report of total size). There is an auxiliary category 'extractors' that will cause all enabled extractors to be loaded, and reports on their status and configuration. [Default: None]
- **format** (*{'native', 'jsonld'}, optional*) – format to use for the 'metadata' result property. 'native' will report the output of extractors as separate metadata properties that are stored under the name of the associated extractor; 'jsonld' composes a JSON-LD graph document, while stripping any information that does not appear to be properly typed linked data (extractor reports no '@context' field). [Default: 'native']
- **on_failure** (*{'ignore', 'continue', 'stop'}, optional*) – behavior to perform on failure: 'ignore' any failure is reported, but does not cause an exception; 'continue' if any failure occurs an exception will be raised at the end, but processing other actions will continue for as long as possible; 'stop': processing will stop on first failure and an exception is raised. A failure is any result with status 'impossible' or 'error'. Raised exception is an IncompleteResultsError that carries the result dictionaries of the failures in its *failed* attribute. [Default: 'continue']
- **proc_post** – Like *proc_pre*, but procedures are executed after the main command has finished. [Default: None]
- **proc_pre** – DataLad procedure to run prior to the main command. The argument a list of lists with procedure names and optional arguments. Procedures are called in the order their are given in this list. It is important to provide the respective target dataset to run a procedure on as the *dataset* argument of the main command. [Default: None]
- **result_filter** (*callable or None, optional*) – if given, each to-be-returned status dictionary is passed to this callable, and is only returned if the callable's return value does not evaluate to False or a ValueError exception is raised. If the given callable supports ***kwargs* it will additionally be passed the keyword arguments of the original API call. [Default: None]
- **result_renderer** (*{'default', 'json', 'json_pp', 'tailored'} or None, optional*) – format of return value rendering on stdout. [Default: None]

- **result_xfm** (*{'datasets', 'successdatasets-or-none', 'paths', 'relpaths', 'metadata'} or callable or None, optional*) – if given, each to-be-returned result status dictionary is passed to this callable, and its return value becomes the result instead. This is different from *result_filter*, as it can perform arbitrary transformation of the result value. This is mostly useful for top- level command invocations that need to provide the results in a particular format. Instead of a callable, a label for a pre-crafted result transformation can be given. [Default: None]
- **return_type** (*{'generator', 'list', 'item-or-list'}, optional*) – return value behavior switch. If 'item-or-list' a single value is returned instead of a one-item return value list, or a list in case of multiple return values. *None* is return in case of an empty list. [Default: 'list']

3.1.2 datalad.api.meta_aggregate

datalad.api.**meta_aggregate**(*path=None*, *dataset=None*, *recursive=False*, *recursion_limit=None*, *into='top'*, *force=None*)

Aggregate metadata of one or more (sub)datasets for later reporting.

Metadata aggregation refers to a procedure that extracts metadata present in a dataset into a portable representation that is stored in a standardized (internal) format. Moreover, metadata aggregation can also extract metadata in this format from one dataset and store it in another (super)dataset. Based on such collections of aggregated metadata it is then possible to discover particular (sub)datasets and individual files in them, without having to obtain the actual dataset repositories first (see the DataLad 'meta-report' command).

To enable aggregation of metadata that are contained in files of a dataset, one has to enable one or more metadata extractor for a dataset. DataLad supports a number of common metadata standards, such as the Exchangeable Image File Format (EXIF), Adobe's Extensible Metadata Platform (XMP), and various audio file metadata systems like ID3. DataLad extension packages can provide metadata data extractors for additional metadata sources. The list of metadata extractors available to a particular DataLad installation is reported by the 'wtf' command ('datalad wtf').

Enabling a metadata extractor for a dataset is done by adding its name to the 'datalad.metadata.nativetype' configuration variable in the dataset's configuration file (.datalad/config), e.g.:

```
[datalad "metadata"]
  nativetype = exif
  nativetype = xmp
```

If an enabled metadata extractor is not available in a particular DataLad installation, metadata extraction will not succeed in order to avoid inconsistent aggregation results.

Enabling multiple extractors is supported. In this case, metadata are extracted by each extractor individually, and stored alongside each other. Metadata aggregation will also extract DataLad's internal metadata ('metalad_core'), and git-annex file metadata ('metalad_annex').

Metadata aggregation can be performed recursively, in order to aggregate all metadata from all subdatasets. By default, re-aggregation of metadata inspects modifications of datasets and metadata extractor parameterization with respect to the last aggregated state. For performance reasons, re-aggregation will be automatically skipped, if no relevant change is detected. This default behavior can be altered via the `--force` argument.

Depending on the versatility of the present metadata and the number of dataset or files, aggregated metadata can grow prohibitively large or take a long time to process. See the documentation of the `extract-metadata` command for a number of configuration settings that can be used to tailor this process on a per-dataset basis.

Parameters

- **path** (*sequence of str or None, optional*) – path to (sub)datasets whose metadata shall be aggregated. When a given path is pointing into a dataset (instead of to its root), the metadata of the containing dataset will be aggregated. [Default: None]
- **dataset** (*Dataset or None, optional*) – topmost dataset metadata will be aggregated into. If no dataset is specified, a datasets will be discovered based on the current working directory. [Default: None]
- **recursive** (*bool, optional*) – if set, recurse into potential subdataset. [Default: False]
- **recursion_limit** (*int or None, optional*) – limit recursion into subdataset to the given number of levels. [Default: None]
- **into** (*{'top', 'all'}, optional*) – which datasets shall receive the aggregated metadata: all datasets from any leaf dataset to the top-level target dataset including all intermediate datasets (all), or just the top-level dataset (top). [Default: 'top']
- **force** (*{'extraction', 'fromscratch', 'ignoreextractorchange', None}, optional*) – Disable specific optimizations: 'extraction' overrides change detection and engages all enabled extractors regardless of whether an actual change in a dataset's state is detected with respect to any existing metadata aggregate; 'fromscratch' wipes out any existing metadata aggregates first, including aggregates for unavailable datasets (implies 'extraction'). 'ignoreextractorchange' disables comparison of current and recorded extractor parametrization and avoids re-extraction due to extractor changes alone. [Default: None]
- **on_failure** (*{'ignore', 'continue', 'stop'}, optional*) – behavior to perform on failure: 'ignore' any failure is reported, but does not cause an exception; 'continue' if any failure occurs an exception will be raised at the end, but processing other actions will continue for as long as possible; 'stop': processing will stop on first failure and an exception is raised. A failure is any result with status 'impossible' or 'error'. Raised exception is an IncompleteResultsError that carries the result dictionaries of the failures in its *failed* attribute. [Default: 'continue']
- **proc_post** – Like *proc_pre*, but procedures are executed after the main command has finished. [Default: None]
- **proc_pre** – DataLad procedure to run prior to the main command. The argument a list of lists with procedure names and optional arguments. Procedures are called in the order their are given in this list. It is important to provide the respective target dataset to run a procedure on as the *dataset* argument of the main command. [Default: None]
- **result_filter** (*callable or None, optional*) – if given, each to-be-returned status dictionary is passed to this callable, and is only returned if the callable's return value does not evaluate to False or a ValueError exception is raised. If the given callable supports ***kwargs* it will additionally be passed the keyword arguments of the original API call. [Default: None]
- **result_renderer** (*{'default', 'json', 'json_pp', 'tailored'} or None, optional*) – format of return value rendering on stdout. [Default: None]
- **result_xfm** (*{'datasets', 'successdatasets-or-none', 'paths', 'relpaths', 'metadata'} or callable or None, optional*) – if given, each to-be-returned result status dictionary is passed to this callable, and its return value becomes the result instead. This is different from *result_filter*, as it can perform arbitrary transformation of the result value. This is mostly useful for top- level command invocations that need to provide the results in a particular format. Instead of a callable, a label for a pre-crafted result transformation can be given. [Default: None]

- **return_type** (*{'generator', 'list', 'item-or-list'}, optional*) – return value behavior switch. If 'item-or-list' a single value is returned instead of a one-item return value list, or a list in case of multiple return values. *None* is return in case of an empty list. [Default: 'list']

3.1.3 datalad.api.meta_report

datalad.api.**meta_report**(*path=None*, *dataset=None*, *reporton='all'*, *recursive=False*)

Query a dataset's aggregated metadata for dataset and file metadata

Two types of metadata are supported:

1. metadata describing a dataset as a whole (dataset-global metadata), and
2. metadata for files in a dataset (content metadata).

Both types can be queried with this command, and a specific type is requested via the *–reporton* argument.

Examples

Report the metadata of a single file, the queried dataset is determined based on the current working directory:

```
% datalad query-metadata somedir/subdir/thisfile.dat
```

Sometimes it is helpful to get metadata records formatted in a more accessible form, here as pretty-printed JSON:

```
% datalad -f json_pp query-metadata somedir/subdir/thisfile.dat
```

Same query as above, but specify which dataset to query (must be containing the query path):

```
% datalad query-metadata -d . somedir/subdir/thisfile.dat
```

Report any metadata record of any dataset known to the queried dataset:

```
% datalad query-metadata --recursive --reporton datasets
```

Get a JSON-formatted report of metadata aggregates in a dataset, incl. information on enabled metadata extractors, dataset versions, dataset IDs, and dataset paths:

```
% datalad -f json query-metadata --reporton aggregates
```

Parameters

- **path** (*sequence of str or None, optional*) – path(s) to query metadata for. [Default: None]
- **dataset** (*Dataset or None, optional*) – dataset to query. If not given, a dataset will be determined based on the current working directory. [Default: None]
- **reporton** (*{'all', 'jsonld', 'datasets', 'files', 'aggregates'}, optional*) – what type of metadata to report on: dataset-global metadata only ('datasets'), metadata on dataset content/files only ('files'), both ('all', default). 'jsonld' is an alternative mode to report all available metadata with JSON-LD markup. A single metadata result with the entire metadata graph matching the query will be reported, all non-JSON-LD-type metadata will be ignored. There is an auxiliary category

'aggregates' that reports on which metadata aggregates are present in the queried dataset. [Default: 'all']

- **recursive** (*bool, optional*) – if set, recursively report on any matching metadata based on given paths or reference dataset. Note, setting this option does not cause any recursion into potential subdatasets on the filesystem. It merely determines what metadata is being reported from the given/discovered reference dataset. [Default: False]
- **on_failure** (*{'ignore', 'continue', 'stop'}, optional*) – behavior to perform on failure: 'ignore' any failure is reported, but does not cause an exception; 'continue' if any failure occurs an exception will be raised at the end, but processing other actions will continue for as long as possible; 'stop': processing will stop on first failure and an exception is raised. A failure is any result with status 'impossible' or 'error'. Raised exception is an IncompleteResultsError that carries the result dictionaries of the failures in its *failed* attribute. [Default: 'continue']
- **proc_post** – Like *proc_pre*, but procedures are executed after the main command has finished. [Default: None]
- **proc_pre** – DataLad procedure to run prior to the main command. The argument a list of lists with procedure names and optional arguments. Procedures are called in the order their are given in this list. It is important to provide the respective target dataset to run a procedure on as the *dataset* argument of the main command. [Default: None]
- **result_filter** (*callable or None, optional*) – if given, each to-be-returned status dictionary is passed to this callable, and is only returned if the callable's return value does not evaluate to False or a ValueError exception is raised. If the given callable supports ***kwargs* it will additionally be passed the keyword arguments of the original API call. [Default: None]
- **result_renderer** (*{'default', 'json', 'json_pp', 'tailored'} or None, optional*) – format of return value rendering on stdout. [Default: None]
- **result_xfm** (*{'datasets', 'successdatasets-or-none', 'paths', 'relpaths', 'metadata'} or callable or None, optional*) – if given, each to-be-returned result status dictionary is passed to this callable, and its return value becomes the result instead. This is different from *result_filter*, as it can perform arbitrary transformation of the result value. This is mostly useful for top- level command invocations that need to provide the results in a particular format. Instead of a callable, a label for a pre-crafted result transformation can be given. [Default: None]
- **return_type** (*{'generator', 'list', 'item-or-list'}, optional*) – return value behavior switch. If 'item-or-list' a single value is returned instead of a one-item return value list, or a list in case of multiple return values. *None* is return in case of an empty list. [Default: 'list']

3.2 Metadata extractors

CHAPTER 4

Acknowledgments

DataLad development is being performed as part of a US-German collaboration in computational neuroscience (CRCNS) project "DataGit: converging catalogues, warehouses, and deployment logistics into a federated 'data distribution'" (Halchenko/Hanke), co-funded by the US National Science Foundation (NSF 1429999) and the German Federal Ministry of Education and Research (BMBF 01GQ1411). Additional support is provided by the German federal state of Saxony-Anhalt and the European Regional Development Fund (ERDF), Project: Center for Behavioral Brain Sciences, Imaging Platform

DataLad is built atop the git-annex software that is being developed and maintained by Joey Hess.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Index

M