How To: HOMER Metadata Crawler - by TUM/NFDI4Ing - with adaptions by InHPC-DE

LRZ

12 February, 2026

Table of Contents

1 What is this about?

For **publishing data** via the publication mechanisms available at the GCS supercomputing centres (and also elsewhere), you will probably need to collect **DataCite-compliant (at least minimal) metadata.**

Some of these metadata are probably already available within your input/output files or within the queuing system.

HOMER allows you to **extract these metadata according to rules and (with a converter) write them into a YAML file complying to the DataCite standard as utilised by the GCS centres.**

Repositories:

- HOMER with adaptations by InHPC-DE[1]
- HOMER to Datacite converter[2]
- Original HOMER[3]

1. https://gitlab-ce.lrz.de/rdm-public-inhpc-lrz/homer-fork
2. https://gitlab-ce.lrz.de/rdm-public-inhpc-lrz/homer-to-ltds-datacite-converter
3. https://gitlab.lrz.de/nfdi4ing/crawler

2 HOMER

2.1 Concept

The **HOMER** crawler (**H**PMC tool for **O**ntology-based **M**etadata **E**xtraction and **R**e-use) was originally developed in the context of the "NFDI4Ing" (National Research Data Infrastructure for Engineering Sciences) at the Chair of Aerodynamics and Fluid Mechanics of Technical University of Munich.

It is capable to read out ontologies (.owl-files) and create a dictionary of relevant properties to be filled by the user. In a second step this dictionary is used to create a metadata-file from user-provided inputs (log files, ...) which accompanies the main dataset. The dictionary therein may be filled with instruction where to find the relevant inputs rather than hard-coding them directly.

2.2 LRZ Fork

A LRZ Fork of Homer, as provided here, has slight modifications resulting in a metadata.yaml compatible with the **GCS centres' standards (for uploads e.g. to B2SHARE or the LRZ FAIR Data Portal)** and can be found in this Gitlab repository[4].

4. https://gitlab-ce.lrz.de/rdm-public-inhpc-lrz/homer-fork

3 Setup HOMER

3.1 Locally

python/pip commands used here refer to python3 and pip3 respectively

First, clone this Gitlab repository[5] to your local machine and navigate there

```
git clone <SSH or HTTPS URL>
cd Homer-fork
```

It is recommended to use a virtual environment. Create it inside the HOMER directory and activate it using the following commands:

```
python -m venv venv
source venv/bin/activate
```

Then install all dependencies and required packages:

```
pip install -r requirements.txt
python setup.py install
```

3.2 HPC-Cluster

This section will soon include a manual on how to setup HOMER on your HPC-Cluster

5. https://gitlab-ce.lrz.de/rdm-public-inhpc-lrz/homer-fork

4 Use HOMER

4.1 **Step 1: Generate a dictionary from an ontology**

We strongly recommend to use the ltds-datacite.owl provided in the repositories 'ltds_datacite_example' directory![6]

For the first step just parse it using the following command:

```
python build/lib/OwlImplementation/ClassUtils.py ltds_datacite_example/ltds-
datacite.owl ltds_datacite_example/steps/step_1.json
```

4.2 **Step 2:** Apply the multiplexer

The following script instructs the multiplexer to multiply and expand the fields based on the empty dictionary defined in Step 1. It creates a new file containing **still empty** metadata fields you will harvest with this default configuration.

```
python build/lib/OwlImplementation/Multiplexer.py ltds_datacite_example/steps/
step_1.json ltds_datacite_example/steps/step_2.json
```

The output file (step_2.json) should look something like this:

6. https://gitlab-ce.lrz.de/rdm-public-inhpc-lrz/homer-fork/-/blob/main/ltds_datacite_example/ltds-datacite.owl

```
"Contributor_1": {···
},
"Contributor_2": {···
},
"Contributor_3": {
    "__restrictions__": [],
    "name": {
        "path": "",
        "type": "",
        "pattern": "",
        "postprocessor": {
            "type": "",
            "args": ""
        }
    },
    "nameType": {
        "path": "",
        "type": "",
        "pattern": "",
        "postprocessor": {
            "type": "",
            "args": ""
        }
    },
    "givenName": {
        "path": "",
        "type": "",
        "pattern": "",
        "postprocessor": {
            "type": "",
            "args": ""
        }
    },
    "familyName": {
        "path": "",
        "type": "",
        "pattern": "",
        "postprocessor": {
            "type": "",
            "args": ""
        }
    },
    "nameIdentifiers_nameIdentifier": {
        "path": "",
        "type": "",
        "pattern": "",
        "postprocessor": {
            "type": "",
            "args": ""
        }
    },
    "nameIdentifiers_schemeURI": {
```

4.3 Step 3: Fill the empty fields with the desired patterns

In Step 2 we defined which fields ('keys' in the dictionary) will be present in the final metadata. Now we need to assign the desired values to each field.

1. Copy **step_2.json** and save it as **step_3.json**
2. Take a look how each field is represented in your input data and **fill in the empty values accordingly**.

Supported methods to automatically extract metadata from files (see original HOMER docs for detailed information[7]):

- plain text ("string")
- Regular Expression pattern ("regex")
- execute shell commands ("os")
- post-process the extracted string using different methods.

field	description
path	Tells HOMER where to extract the specific field value from. Needed only for "type": "regex"
type	Either "string", "regex" or "os"
pattern	"string": plain text string you want to enter "regex": regex expression being used for extraction from "path" "os": some shell command
postprocessor	Postprocess strings using one of the foloowing processors: "min", "median", "max", "stddev", "mean", "var", "sqrt", "log2", "log10", "sort", "regex", "stringstrip"

Below is an example on how to extract all details about "Contributor_3" from a logfile (testfile.log) and another file holding additional information (address_janedoe.txt):

7. https://gitlab.lrz.de/nfdi4ing/crawler/-/blob/master/Documentation/Documentation_HOMER_V_0_1.pdf?ref_type=heads

step_4.json U

TempDir > step_4.json > Contributor_1 > nameType > postprocessor

```
    },
    "Contributor_3": {
        "__restrictions__": [],
        "name": {
            "path": "TempDir/data/testfile.log",
            "type": "regex",
            "pattern": "Author2: (.*)",
            "postprocessor": {
                "type": "",
                "args": ""
            }
        },
        "nameType": { ···
        },
        "givenName": { ···
        },
        "familyName": {
            "path": "TempDir/data/testfile.log",
            "type": "regex",
            "pattern": "Author2: .* (.*)",
            "postprocessor": {
                "type": "",
                "args": ""
            }
        },
        "nameIdentifiers_nameIdentifier": {
            "path": "TempDir/data/address_janedoe.txt",
            "type": "regex",
            "pattern": ".* (https://orcid.*)",
            "postprocessor": {
                "type": "",
                "args": ""
            }
        },
        "nameIdentifiers_schemeURI": { ···
        },
        "nameIdentifiers_nameIdentifierScheme": { ···
        },
        "affiliations_name": {
            "path": "",
            "type": "os",
            "pattern": "awk 'BEGIN{oldrec1=\"\"; oldrec2=\"\" ; count=0} {if (oldrec1==\"Jane\" && oldrec2==\"Doe\") {if (count<=1) {pr
            "postprocessor": {
                "type": "stringstrip",
                "args": ""
            }
        },
        "affiliations_affiliationIdentifier": {
            "path": "",
            "type": "os",
            "pattern": "awk 'BEGIN{count=0} {if ($1==\"ROR:\") {if (count<=1) {print $2}; count=count+1}}' ./TempDir/data/address_jane.
            "postprocessor": {
                "type": "stringstrip",
```

address_janedoe.txt U

TempDir > data > address_janedoe.txt

```
Jane Doe
Technische Universität München
Arcisstraße 21
80333 München
ROR: https://ror.org/02kkvpp62
ORCID: https://orcid.org/0000-0001-8888-3333
e-mail: janedoe@tum.de
Tel.: +49-89-54321-3215

Jane Doe
Leibniz-Rechenzentrum
Boltzmannstr. 1
85748 Garching b. München
ROR: https://ror.org/05558nw16
ORCID: https://orcid.org/0000-0001-8888-3333
e-mail: janedoe@tum.de
Tel.: +49-89-54321-3215
```

testfile.log

TempDir > data > testfile.log

```
opening input files...
set numthreads:            6

---------------------------------
Monte Carlo RT Code for Supernovae
Author1: Max Mustermann
Author2: Jane Doe
Code Run By: John Doe

2024-01-17
---------------------------------

[SOME ARBITRARY OUTPUT]
```

4.4 Step 4: Generate Metadata Object from input data

Finally execute the Crawler, which fetches metadata based on the parameters specified in the previous steps from your input data (logfiles, etc.). It then creates an output file in the .yaml format containing your metadata!

```
python build/lib/OwlImplementation/EntityUtils.py ltds_datacite_example/steps/
step_3.json ltds_datacite_example/homer_metadata.yaml
```

The resulting .yaml file should look something like this:


```
"Contributor_1":
    "name": "John Doe"
    "nameType": "Personal"
    "givenName": "John"
    "familyName": "Doe"
    "nameIdentifiers_nameIdentifier": "https://orcid.org/0000-0001-8342-1479"
    "nameIdentifiers_schemeURI": "https://orcid.org"
    "nameIdentifiers_nameIdentifierScheme": "ORCID"
    "affiliations_name": "Leibniz-Rechenzentrum"
    "affiliations_affiliationIdentifier": "https://ror.org/05558nw16"
    "affiliations_affiliationIdentifierScheme": "ROR"
    "contributorType": "Researcher"
"Contributor_2":
    "name": "Max Mustermann"
    "nameType": "Personal"
    "givenName": "Max"
    "familyName": "Mustermann"
    "nameIdentifiers_nameIdentifier": "https://orcid.org/0000-0001-8551-1402"
    "nameIdentifiers_schemeURI": "https://orcid.org"
    "nameIdentifiers_nameIdentifierScheme": "ORCID"
    "affiliations_name": "Technische Universität München"
    "affiliations_affiliationIdentifier": "https://ror.org/02kkvpp62"
    "affiliations_affiliationIdentifierScheme": "ROR"
    "contributorType": ""
"Contributor_3":
    "name": "Jane Doe"
    "nameType": "Personal"
    "givenName": "Jane"
    "familyName": "Doe"
    "nameIdentifiers_nameIdentifier": "https://orcid.org/0000-0001-8888-3333"
    "nameIdentifiers_schemeURI": "https://orcid.org"
    "nameIdentifiers_nameIdentifierScheme": "ORCID"
    "affiliations_name": "Technische Universität München"
    "affiliations_affiliationIdentifier": "https://ror.org/02kkvpp62"
    "affiliations_affiliationIdentifierScheme": "ROR"
    "contributorType": "ProjectManager"
```

4.5 (optional) Transform metadata into Datacite standard

Converts the native HOMER metadata into a datacite compliant metadata file which is compatible with the GCS centres' standards (for uploads e.g. to **B2SHARE**[8]or the **LRZ FAIR Data Portal**[9]). You can find the converter module in this gitlab repo[10]

After copying your file inside the **homer-to-datacite-converter** repo, execute the following command:

```
python homer_to_ltds_datacite.py homer_metadata.yaml datacite_metadata.yaml
```

8. https://b2share.eudat.eu/
9. https://rdm.lab.lrz.de/
10. https://gitlab-ce.lrz.de/rdm-public-inhpc-lrz/homer-to-ltds-datacite-converter