
PyRDM Documentation

Release 0.3

**Christian T. Jacobs
Alexandros Avdis
Gerard J. Gorman
Matthew D. Piggott**

February 02, 2017

1	Introduction	3
1.1	Overview	3
1.2	Licensing	4
2	Getting started	5
2.1	System requirements	5
2.2	Downloading and installing	5
2.3	Configuring	6
2.4	Testing	8
3	PyRDM functionality	9
3.1	Publishing software	9
3.2	Publishing data	9
4	Application: Fluidity-Publish	11
4.1	Enable publishing	11
4.2	Using Fluidity-Publish	11
5	Application: PyRDM-Publish	15
6	References	17
6.1	Journal articles	17
6.2	Conference papers	17
6.3	Posters and presentations	17

Contents:

Introduction

1.1 Overview

PyRDM is a project that focuses on the relationship between data and the scientific software that has either created that data, or operates on that data to create new information. Two examples from geoscience are given for illustrative purposes:

- Dispersion of saline solution into an estuary from desalination plant.
 - Input data: Bathymetry, atmospheric forcing, etc.
 - Software: Pre-/post-processing codes; numerical models.
 - Output data: Provenance; simulated flow fields; diagnostic quantities.
- Characterisation of pores media for carbon sequestration.
 - Input data: Micro-CT images of rock samples.
 - Software: Pre-/post-processing of data (e.g. image segmentation, mesh generation); numerical models.
 - Output data: Provenance; computational meshes that could be used by other flow solvers; computed flow fields; diagnostics.

From these two examples one can already start to see a great deal of generality emerging. In order to achieve reproducibility, or indeed computability, one must be able to capture all of the input data (i.e. collected/measured data which should also have its associated provenance). Next the actual software that draws information from that data must be captured, and care must be taken to ensure that the same version that created a particular result from the data is used. Finally, the output data from the software must be captured, including provenance data that details how the input data and software were used to create this new data.

The PyRDM library aims to address these needs by facilitating the automated management and publication of software source code and data via online, citable repositories hosted by [Figshare](#), [Zenodo](#), or a [DSpace](#)-based service. The library can be readily incorporated into scientific workflows to allow research outputs to be curated and shared in a straight-forward manner. Further details and technical information are provided in the following paper:

- **C. T. Jacobs, A. Avdis, G. J. Gorman, M. D. Piggott (2014).** *PyRDM: A Python-based library for automating the management and online publication of scientific software and data.* Journal of Open Research Software, 2(1):e28, DOI: [10.5334/jors.bj](https://doi.org/10.5334/jors.bj)

and in the resources listed [here](#).

PyRDM is open-source, and is available to download from the project's GitHub repository: <https://github.com/pyrdm/pyrdm>

1.2 Licensing

PyRDM is released under the GNU General Public License. Further details can be found in the COPYING file supplied with this software.

Getting started

2.1 System requirements

A standard Python installation is required, as well as any additional Python modules that are listed in the [README file](#) under the “Dependencies” section. PyRDM is designed to run on the Linux operating system.

It is recommended that users use the terminal to install and run PyRDM.

2.2 Downloading and installing

PyRDM can either be installed from source or through a [Conda package](#), both of which are described below.

2.2.1 From source

PyRDM’s source code is hosted on GitHub and can be found here: <https://github.com/pyrdm/pyrdm>. The first step is to download the source code using:

```
git clone https://github.com/pyrdm/pyrdm.git pyrdm
```

The core dependencies that PyRDM needs to function can then be installed by navigating to the base directory of PyRDM (i.e. the directory that the Makefile is in) using

```
cd pyrdm
```

and executing

```
sudo pip install -r requirements.txt
```

Use the following command to install the PyRDM library:

```
sudo make install
```

Note 1: `sudo` is likely to be necessary here if the default install directory is located outside of `/home`. This will yield a system-wide install of PyRDM, which is recommended.

Note 2: In order for Python to find the PyRDM module, you will need to add the PyRDM base directory to your `PYTHONPATH` environment variable, unless you have used `sudo` as mentioned in Note 1 above. This can be achieved using:

```
export PYTHONPATH=$PYTHONPATH:/path/to/pyrdm
```

You may wish to add this statement to your `/home/your_username/.bashrc` or `/etc/bash.bashrc` files so the `PYTHONPATH` is set correctly each time you log in.

2.2.2 Conda package

Users of Conda can install PyRDM and its dependencies using

```
conda install -c ctjacobs -c pypi -c auto -c ioos -c conda-forge pyrdm
```

2.3 Configuring

You should copy the contents of the file `pyrdm.ini.example` to a new file called `pyrdm.ini` and save it in the `/home/your_username/.config` directory. If this directory does not exist, please create it first using

```
mkdir /home/your_username/.config
```

The contents of the new file `pyrdm.ini` should then be modified as per the guidance in the following subsections.

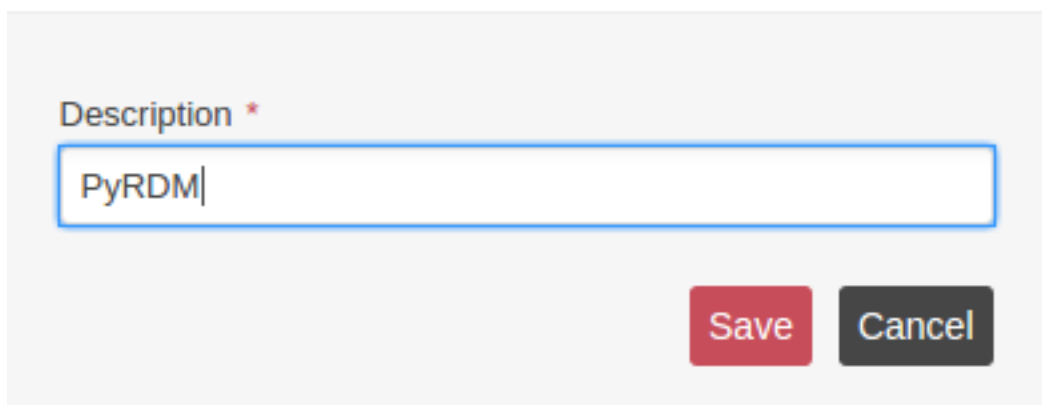
2.3.1 Figshare authentication

PyRDM requires a personal authentication token in order to publish and modify files using your Figshare account. You will need to login and use the Figshare web interface to generate this authentication token, after which you should paste it into the `figshare` section of the configuration file.

1. Go to <http://figshare.com/account/applications>
2. Click `Create Personal Token`
3. In the description box, type “PyRDM” as per *figure:create_token*.

Create a new personal token

You can use this to access our API without going through the 3 legged oauth process.



The image shows a web form for creating a personal access token. It features a label 'Description *' above a text input field. The input field contains the text 'PyRDM|'. Below the input field, there are two buttons: a red 'Save' button and a dark grey 'Cancel' button.

Fig. 2.1: The details for the new personal access token.

4. Click **Save**. The token will appear and should be pasted into the `pyrdm.ini` configuration file.
5. Click **Done** and the new token should appear in the list, as per *figure:token_list*.

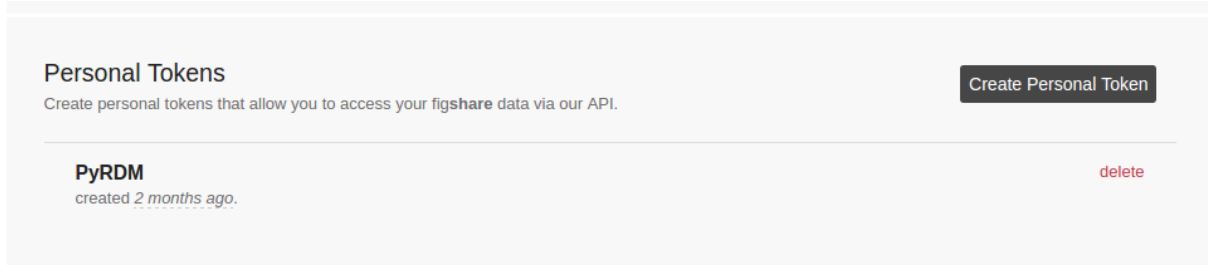


Fig. 2.2: The new personal access token.

Note: If you are publishing through a group account, you will need to ask the account’s administrator for the authentication details.

2.3.2 Zenodo authentication

Zenodo uses a personal access token to handle authentication.

1. Go to <http://zenodo.org/account/settings/applications/tokens/new/>
2. Enter PyRDM as the name of the token. Ensure that `deposit:actions` and `deposit:write` are selected, as per *figure:zenodo_token*.

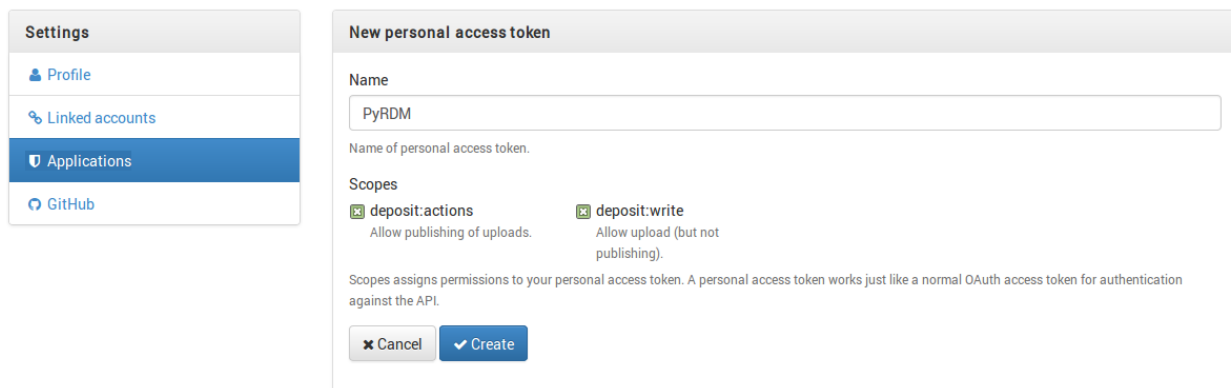


Fig. 2.3: Setting up a new personal access token for PyRDM.

3. Click **Create**. The access token should be pasted into the `[zenodo]` section of the `pyrdm.ini` configuration file.
4. Zenodo requires at least one author’s name and affiliation to be present when creating a deposition. For a software deposition, PyRDM will try to obtain this information from the `AUTHORS` file. However, for a dataset deposition, this information needs to be provided under the `[general]` section of the `pyrdm.ini` configuration file.

2.3.3 DSpace authentication

PyRDM provides limited support for publishing with DSpace-based services which use the [SWORD protocol](#) (version 2).

1. Locate the URL of the SWORD Service Document, and add it under the `[dspace]` section of the `pyrdm.ini` configuration file. Note: this URL may need to be obtained from the system administrator.
2. Add the title of the DSpace Collection that you want to publish in. Note: the publication's privacy settings are determined by the DSpace Collection, not by the `private` parameter in PyRDM.
3. Add your user name and password used to access the DSpace server. Note: this is currently stored in plain text, so make sure that the PyRDM configuration file is not readable by other users.

2.4 Testing

PyRDM comes with a suite of unit tests which verify the correctness of its functionality. It is recommended that you run these unit tests before using PyRDM by executing:

```
make test
```

on the command line. Many of these tests require access to a Figshare and a Zenodo account, so please ensure that the `pyrdm.ini` setup file contains valid authentication tokens.

PyRDM functionality

PyRDM treats the publication of software source code and data separately, and this section is therefore split into two sub-sections. However, regardless of what is being published, PyRDM will print out the DOI or Handle associated with the online repository, which can be used to formally and properly cite the software and data.

3.1 Publishing software

The publication of software is handled by the `publish_software` method in the `Publication` class. This requires:

- The service (e.g. Figshare) authentication details (see the section on Configuration).
- The software's name.
- The location of the software's Git repository (or the location of any file within that repository) on your local hard drive.
- Optionally, the version of the software that you would like to publish (for Git repositories, this is the SHA-1 commit hash). If this is not provided, PyRDM will publish the `HEAD` of the local Git repository.

3.1.1 Author attribution

If an `AUTHORS` file is provided in the Git repository's base directory, PyRDM parses it and looks for strings of a particular form. At the moment, this form depends on the service being used. However, PyRDM will hopefully be able to use a more standardised way of identifying authors in the future.

- For Figshare, PyRDM looks for `<figshare:xxxx>`, where `xxxx` is a Figshare author ID. This should be specified after each author's full name. An example is:

```
Christian Jacobs <figshare:554577>
```

- For Zenodo, PyRDM looks for `<zenodo: (xxxx; yyyy) >`, where `xxxx` is the author's full name, and `yyyy` is the author's affiliation.
- For DSpace, PyRDM looks for `<dspace:xxxx>`, where `xxxx` is the author's user name on the DSpace server.

PyRDM automatically adds all authors who provide their author information to the software publication.

3.2 Publishing data

The publication of software is handled by the `publish_software` method in the `Publication` class. This requires:

- The service authentication details (see the section on Configuration).
- A dictionary of parameters, containing the following key-value pairs:
 - `title`: the title of the dataset.
 - `description`: a description of the dataset.
 - `files`: a list of paths to the files within the dataset.
 - `category`: the name of a category in [Figshare's categories list](#). This is not required when using Zenodo or DSpace.
 - `tag_name`: a single string, or list of strings, to tag the data's fileset with.
- Optionally, a `pid` (publication ID) if the dataset already exists and you wish to update it. By default, this is set to `None`.

3.2.1 MD5 cross-checks

When a data file is published, the file's MD5 checksum is stored in a corresponding checksum file. The next time the user tries to publish the file, its MD5 checksum is recomputed and compared against the MD5 checksum stored in its corresponding MD5 file. If the two MD5 checksums are different (or the checksum file does not exist), the file is uploaded to the server and the checksum file is updated with the new checksum. If the two checksums are the same, the file is unmodified and is not re-uploaded. This can help prevent unnecessary bandwidth usage and is particularly useful when you have large data files which are not frequently modified.

Application: Fluidity-Publish

Fluidity-Publish is a Python program which uses the PyRDM library. It is designed specifically for use with a computational fluid dynamics code called [Fluidity](#). Currently, you must use the `publishing` branch of Fluidity which contains the publishing functionality:

1. `git clone https://github.com/FluidityProject/fluidity.git`
2. `cd fluidity`
3. `git checkout publishing`

Note that you must build Fluidity (or at least satisfy the `libspud` dependency noted in the README file) before using Fluidity-Publish. You may need to add Fluidity's `'python'` directory to your `PYTHONPATH` environment variable in order for the `libspud` module to be found.

4.1 Enable publishing

If we want to publish the Fluidity source code used to run a particular simulation, along with the input and output data, we first need to enable the `publish` option in that simulation's configuration/options file (the `".fml"` file), as shown in [figure:diamond](#). Simply select which publishing service you wish to use (e.g. Figshare or Zenodo), and under the `input_files` and `output_files` sub-options (see Figure [fig:diamond]), enter the paths (relative to the options file) to the input and output files you wish to publish in the following format (a Python list of strings):

```
["path/to/file1", "path/to/file2", "path/to/file3"]
```

You may use wildcard characters here (e.g. `"*vtu"` to publish all VTK-based simulation output). Instead of creating a new code repository or fileset on Figshare or Zenodo, you may also publish to an existing one by entering its ID in the following option(s):

```
/publish/{software,input_data,output_data}/pid
```

4.2 Using Fluidity-Publish

To publish, you will need to use the `fluidity-publish` program (located in PyRDM's `'bin'` directory) at the command line; this expects one or more of the following options, followed by the (relative or absolute) path to the simulation's configuration file:

- `-s` : Publish the Fluidity source code. This can be used in conjunction with the `-v` option to explicitly choose the version of Fluidity (in the form of the Git SHA-1 hash of a particular commit) that you want to publish.

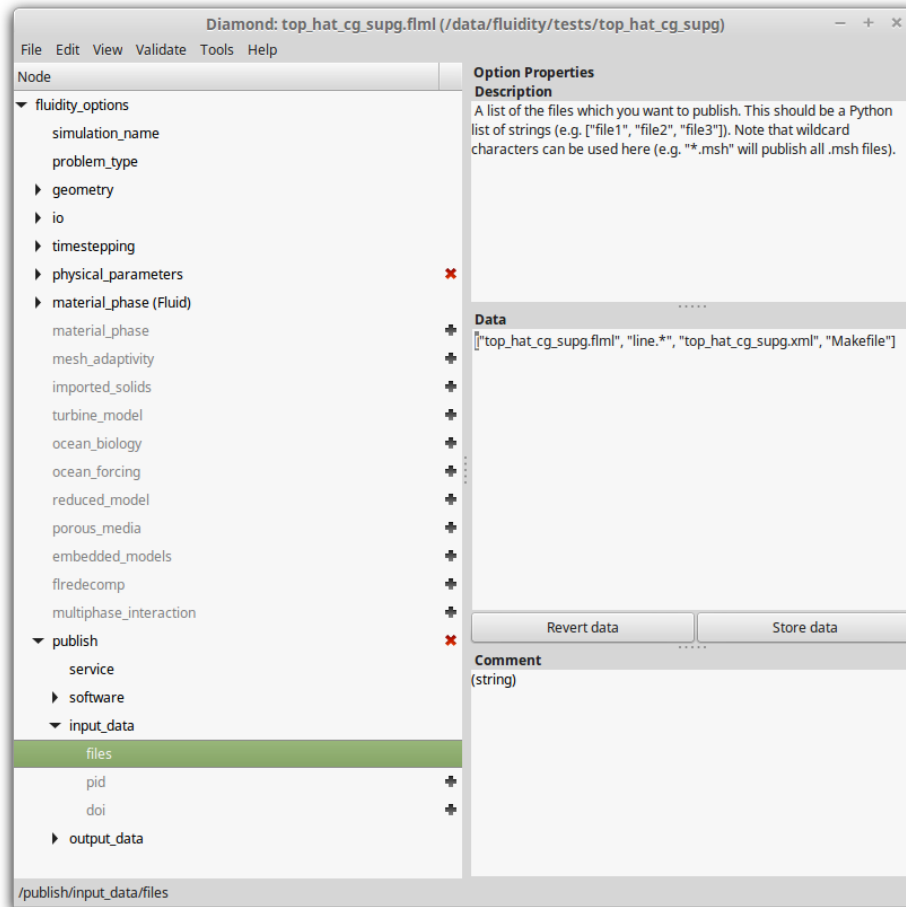


Fig. 4.1: The `publish` option enabled in a simulation's configuration file. The file is being modified in Diamond.

- `-i` : Publish the input data files whose paths are specified in the options file.
- `-o` : Publish the output data files whose paths are specified in the options file.
- `-p` : Publish the software or data, but keep it private. Must be used in conjunction with the `-s`, `-i` or `-o` option. Note that any DOI generated will not be valid until the publication is made public.
- `-l` : Set the log verbosity level (choose ‘critical’, ‘error’, ‘warning’, ‘info’, or ‘debug’).

e.g. `fluidity-publish -i /data/fluidity/tests/top_hat_cg_supg/top_hat_cg_supg.flml`

or, if you did not install PyRDM, change directory (`cd`) to the PyRDM base directory

(e.g. `cd /home/my_username_here/pyrdm`) and use:

```
python bin/fluidity-publish -i /data/fluidity/tests/top_hat_cg_supg/top_hat_cg_supg.flml
```

Note that the software, input data and output data must be published separately. You cannot yet use the `-s`, `-i` and `-o` options together.

Once the publication process has finished, the ID and DOI of the publication will be added to the simulation’s configuration file for future reference. If you wish to publish the simulation data again, the ID and DOI will be re-used (unless you remove them from the configuration file).

4.2.1 Software version

Unless you provide a particular version of Fluidity at the command-line using the `-v` option, Fluidity-Publish will automatically obtain the version of Fluidity from the file `version.h` stored in the `include` directory of the local Fluidity repository on your computer. This file is created at compile-time when the Fluidity binary is built. If this file is not present (perhaps because you haven’t built Fluidity yet), then Fluidity-Publish will instead use the version (SHA-1 key) of the HEAD commit of the local repository.

4.2.2 Provenance data

Fluidity writes a limited amount of provenance data to the header of the simulation’s ‘stat’ file. If you choose to publish the output data (which should include the ‘stat’ file) using the `-o` option, then Fluidity-Publish will (if available) retrieve the IDs and DOIs of the recently published software and input data from the simulation’s options file. It will then add those to the existing provenance data before publishing the output data files that you have specified.

Application: PyRDM-Publish

Another publishing tool (see `bin/pyrdm-publish`) has been created to publish PyRDM itself to Figshare. At the moment this is only used as an example program to demonstrate PyRDM's functionality without the need to install `libspud` (or `Fluidity` and `libspud`) as a dependency, and therefore any Figshare publications produced using this tool will be kept private in your Figshare account. You can visit the “My data” page on Figshare to view the publication details.

Only the source code hosted on PyRDM's GitHub repository can be published; the publishing tool does not publish any input/output files. To use `pyrdm-publish`, you will need to provide the path to the local Git repository on your computer containing PyRDM. For example,

```
pyrdm-publish /home/my_username_here/pyrdm
```

(or `python /path/to/pyrdm/bin/pyrdm-publish /path/to/pyrdm`, if you did not install PyRDM).

The following options are also available:

- `-v` : Publish a specific version (Git SHA-1 hash) of the PyRDM source code.
- `-l` : Set the log verbosity level (choose ‘critical’, ‘error’, ‘warning’, ‘info’, or ‘debug’).

References

6.1 Journal articles

- **C. T. Jacobs, A. Avdis (2016).** *Git-RDM: A research data management plugin for the Git version control system.* The Journal of Open Source Software, 1(2), DOI: [10.21105/joss.00029](https://doi.org/10.21105/joss.00029)
- **C. T. Jacobs, A. Avdis, G. J. Gorman, M. D. Piggott (2014).** *PyRDM: A Python-based library for automating the management and online publication of scientific software and data.* Journal of Open Research Software, 2(1):e28, DOI: [10.5334/jors.bj](https://doi.org/10.5334/jors.bj)

6.2 Conference papers

- **S. L. Mouradian, A. Avdis, M. D. Piggott, C. T. Jacobs, C. Villaret, D. R. de Mijolla, J. Lietava (2016).** *TELEMAC model archive: Integrating open-source tools for the management and visualisation of model data.* In Proceedings of the 23rd TELEMAC-MASCARET User Club, held in Paris, France on 11–13 October 2016. Pre-print: <http://eprints.soton.ac.uk/405307/>
- **C. T. Jacobs, A. Avdis, S. L. Mouradian, M. D. Piggott (2015).** *Integrating Research Data Management into Geographical Information Systems.* In Proceedings of the 5th International Workshop on Semantic Digital Archives, held in Poznań, Poland on 18 September 2015. Pre-print: <http://arxiv.org/abs/1509.04729>

6.3 Posters and presentations

- **C. T. Jacobs, A. Avdis, G. J. Gorman, M. D. Piggott (2015).** *PyRDM: A library to facilitate the automated publication of software and data in computational science.* Presented at the 10th International Digital Curation Conference. DOI: [10.6084/m9.figshare.1318710](https://doi.org/10.6084/m9.figshare.1318710)
- **C. T. Jacobs, A. Avdis, G. J. Gorman, M. D. Piggott (2014).** *RDM Green Shoots Project Report: “Research data management: Where software meets data”.* DOI: [10.6084/m9.figshare.1269127](https://doi.org/10.6084/m9.figshare.1269127)
- **C. T. Jacobs, A. Avdis, G. J. Gorman, M. D. Piggott (2014).** *Research Data Management for Computational Science.* DOI: [10.6084/m9.figshare.1047219](https://doi.org/10.6084/m9.figshare.1047219)