
OMERO upload guide Documentation

Release 0.1.0

Open Microscopy Environment

May 10, 2023

Contents

1	Import Image data	3
2	Change image rendering settings and channel names using the Command Line Interface (CLI)	21
3	Import metadata using the Command Line Interface (CLI)	23
4	Import metadata using the Populate Metadata script in OMERO.web	27
5	Contribute	33

This section shows how to import data using the Desktop client, the Command Line Interface (CLI) and OMERO.dropbox. It also covers advanced import features and post-import processing steps used mainly in the context of the Image Data Resource (IDR).

Contents:

This section shows how to import data either using the Java Desktop client or the Command Line Interface (CLI).

Contents:

1.1 Import data using the Desktop Client

1.1.1 Description

In the first part, we first show how to import data by yourself and for yourself into OMERO using various import strategies. This will be mainly done using the OMERO.insight desktop client.

Second part of this import section will show how to import data for another user, using OMERO.insight. The user importing the data needs to have some admin or restricted-admin privileges. More information about restricted privileges can be found at <https://docs.openmicroscopy.org/latest/omero/sysadmins/restricted-admins.html>

The import for another user requires that the user doing the import has specific privileges. We will use the user importer1, this could be, for example, a facility manager.

We will show:

- How to install the OMERO.insight desktop client on Windows, Mac and Linux.
- How to import data for the user logged in using OMERO.insight.
- How to select a target Project and Dataset or create a new ones in OMERO for the imports.
- How to add Tags to imported images at the import stage, to facilitate the management of these images later in OMERO.server.
- How to import data for other users in OMERO.insight.

1.1.2 Setup

OMERO.insight desktop client installation instructions:.

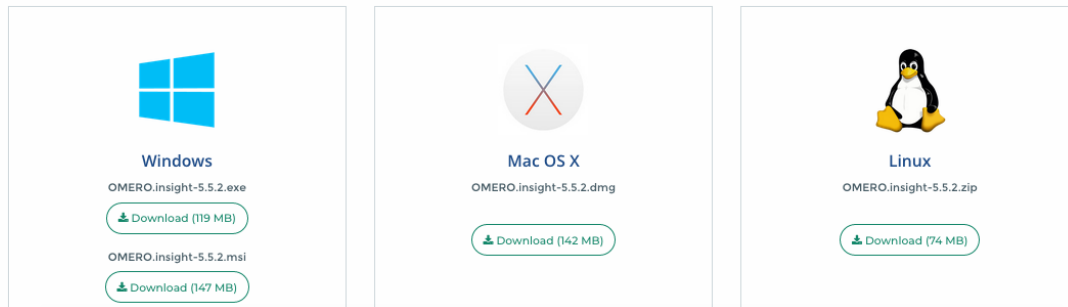
Download the OMERO.insight client corresponding to your operating system at: <https://www.openmicroscopy.org/omero/downloads>



OMERO clients

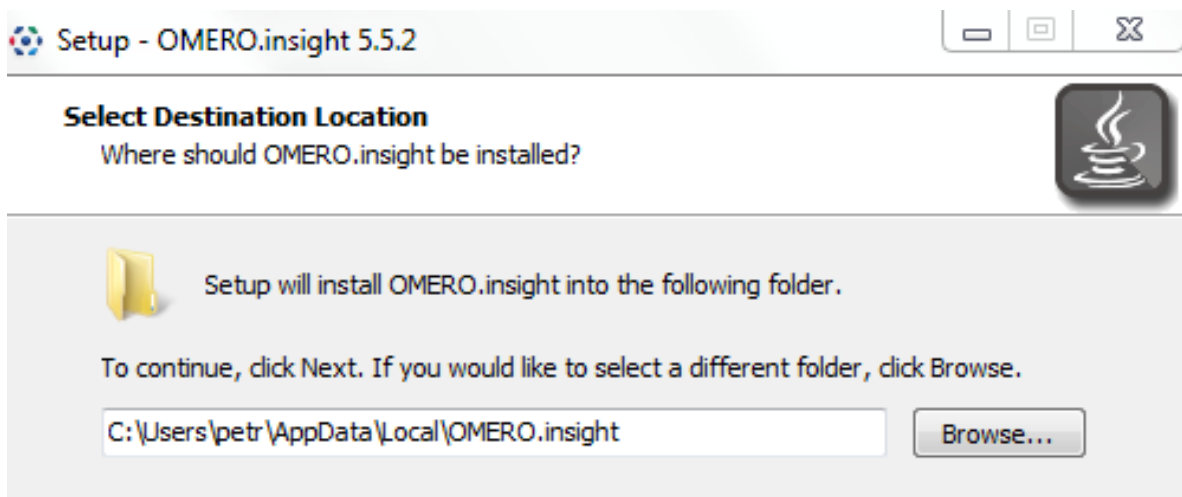
A standard OMERO user just needs to download the client package with the same major version as their institutional server (e.g. 5.3.0 clients will connect to 5.3.5 servers but not to 5.4.0 servers). Full instructions for installation are on the [help site](#).

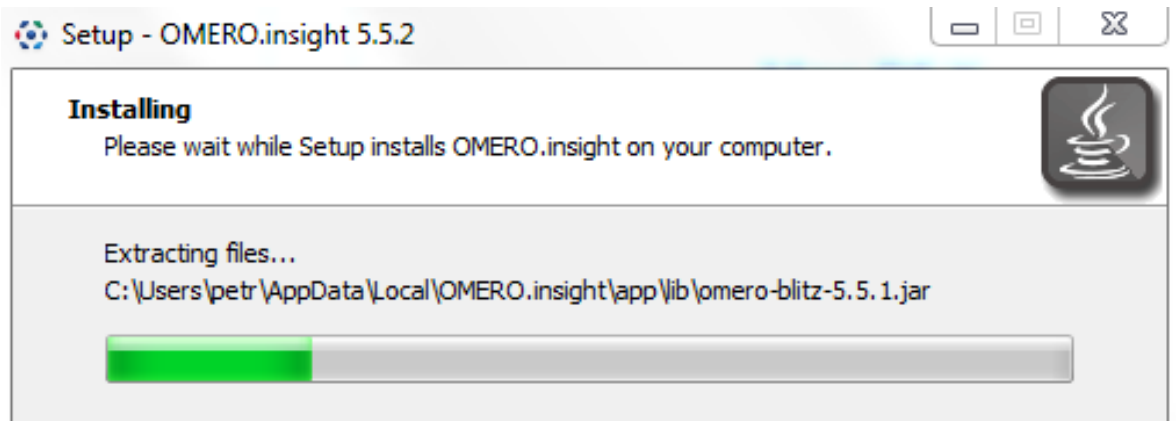
If you do not have an institutional server, you can [apply for an account on our demo server](#).



Windows

- From version 5.5.0, OMERO.insight comes with two installers: .msi and .exe.
- Click onto the downloaded .exe or .msi file.
- This will run the installer. The .exe file installs by default in the userspace apps folder. This can be changed during the installation process.

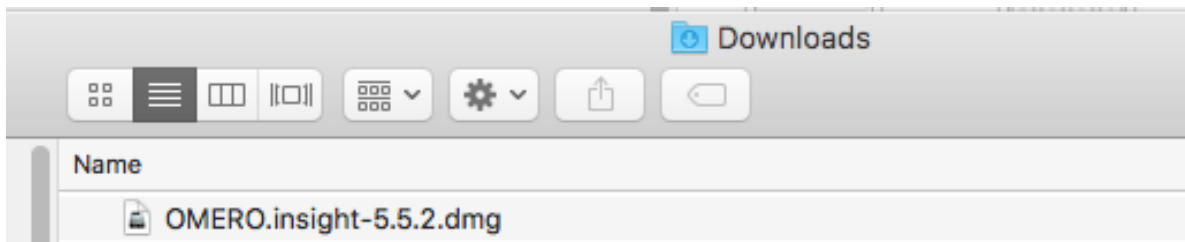




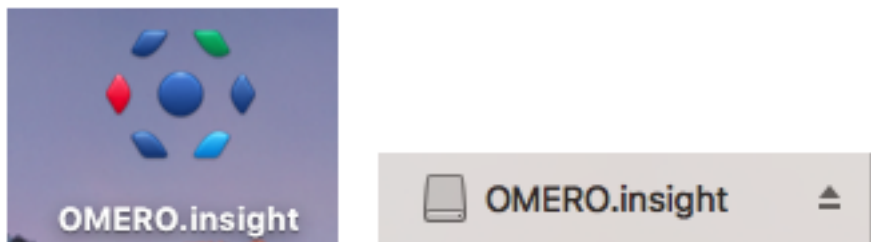
- The .msi installer will deploy the application in the Program Files folder of Windows. *OMERO.insight* is then available to all the users of the target machine. *OMERO.insight* installed in the userspace is only available to the user who did the installation. A Desktop icon and a new *OMERO.insight* Start menu item will be created.

Mac

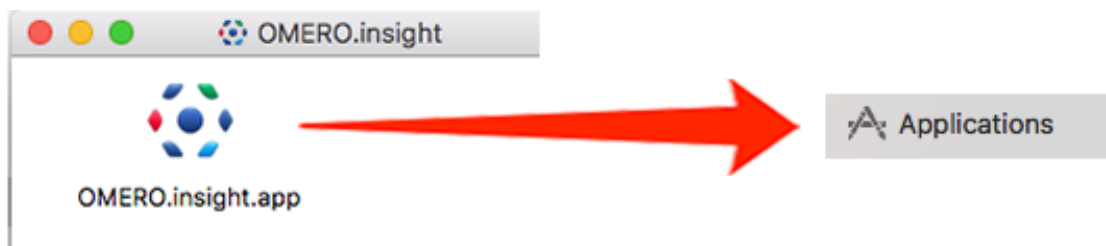
- From version 5.5.0, OMERO.insight can be installed using an Apple DMG (.dmg) file.
- Mac OS X: Click onto the downloaded .dmg installer to start the installation.



- This will mount the DMG to your Mac. The DMG mounts in two places: on your Desktop and in the Finder sidebar under your hard drive.

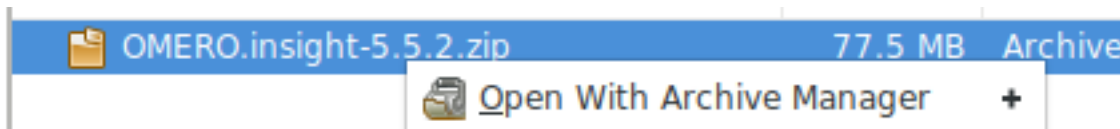


- Clicking either one of these opens the DMG file. When you open a DMG file, you will usually see two things:
 - the app itself.
 - a link to your applications folder.
- Depending on your settings, the Applications folder icon might not appear. In such case, drop the app icon into the Applications folder.

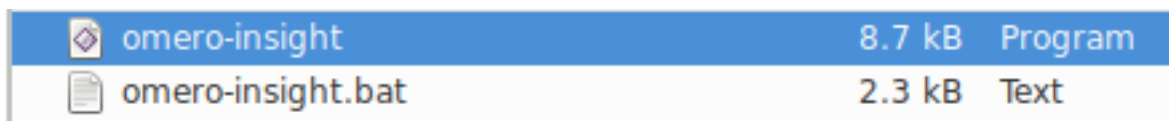


Linux


- Unzip the downloaded .zip file.

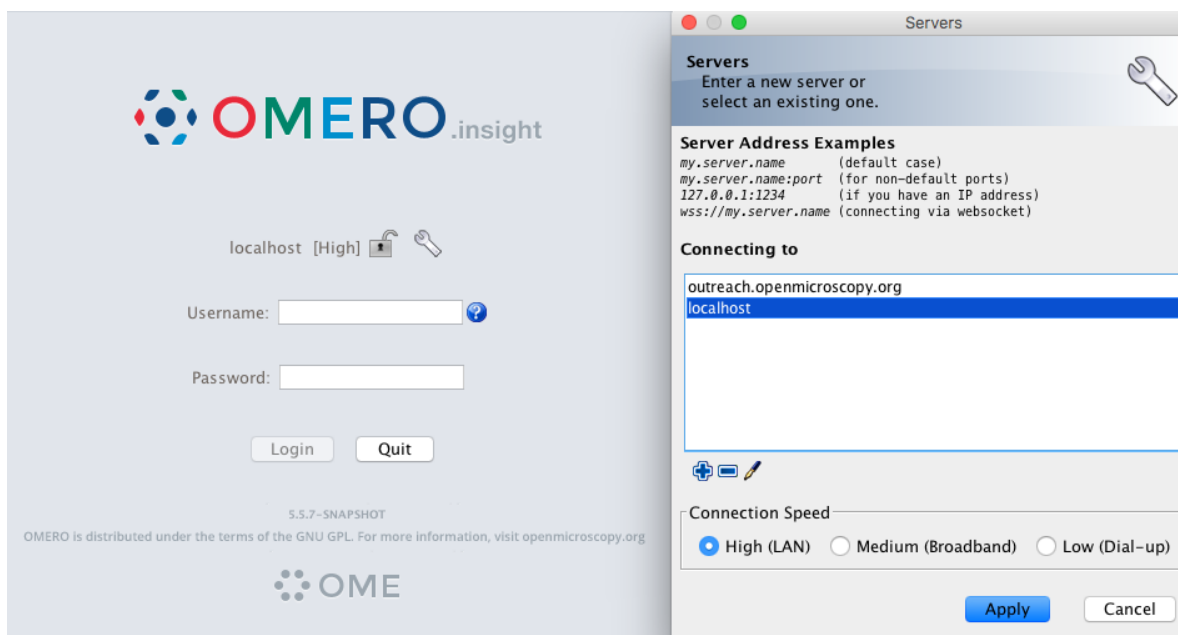


- Click on the omero-insight file to start the application.








1.1.3 Step-by-step

1. Open OMERO.insight and in the login dialog, click on the wrench icon .
2. This will open a list of servers to which you can connect. By default, only “localhost” is listed. Click on the plus icon to add a new line to the list and type into the line the server address.



Note: If your server is running on a default port (4064), as is usually the case, then you can simply just type in the server name such as `my.server.name`. For servers running on a non-default port, e.g. 54064, type the

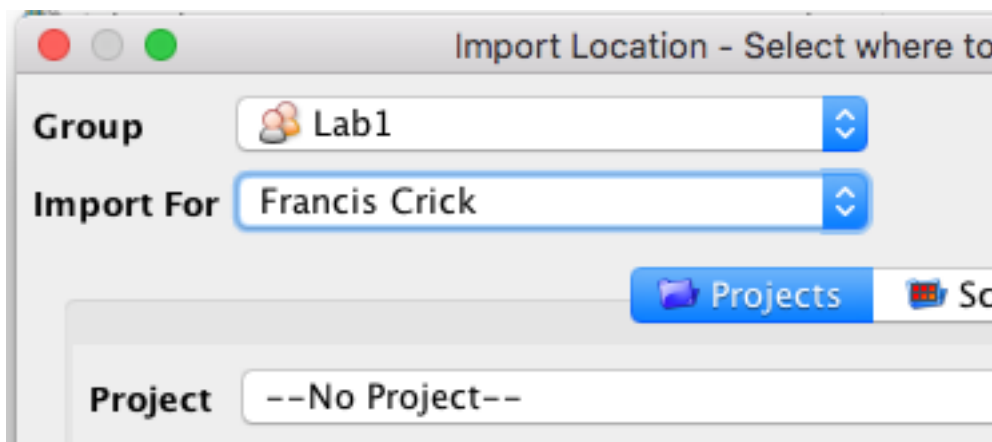
address of the server as `my.server.name:54064` into the above dialog. Alternatively, you can also type in the IP address of your server, or connect using websockets.

3. When done, click *Apply*.
4. Log in using the username and password provided.
5. In OMERO.insight, click on the Importer Icon  in the toolbar.
6. Browse your local hierarchy in the left-hand pane of the importer, select single images or whole folders and add these to the Queue by clicking on the arrow  icon.
7. In the Import Location window, select the target Project and Dataset (existing or create a new one) to import to.
8. Note: If no Dataset is selected or created, a new Dataset will be automatically created and named after the folder containing the images to be imported.
9. Optional: Go to the *Options* tab
 - Click on **Add tag to images**  : to bring the Tag selection dialog.
 - Select the tag(s) on the left-hand side or create a new one.
 - Click  to move the tag(s) to the right-hand side.
 - Click Save.
10. Click on the Import button in the bottom-right corner of the Importer window. You should see two progress bars for every image imported, Upload and Processing.
11. Note: The import of the next image in the queue starts immediately after the Upload of the previous one is finished. The Processing phase of the import is done on the OMERO.server only, and can be finished while the next image(s) is/are being uploaded to the server.
12. Once imports are finished, go back to the OMERO.insight main window and click the Refresh button  above the right-hand pane. This will display the imported images inside the Dataset and/or Project you specified previously in the Import Location window.
13. (demo only step) Now, the demonstrator will log out from OMERO.insight and log in again, this time as some other user and will show the import process again, this time importing a different set of images. After this import, the two sets of images (belonging to two different users) will be shown in the webclient.

1.2 Import for another user

In this example, we show how to import data for another user. A facility manager `importer1` with restricted admin privileges imports the data for user-1. The facility manager has been given the ability to import for others.

The steps are the same as before for the normal import, but as `importer1` has the permission to import for another user there are two drop down menus for selecting the user and group to import for:



- Select Group: 'Lab1'
- Select User: 'Francis Crick'
- Continue the import workflow as usual.

A restriction of OMERO.insight is that the user `importer1` needs to be a member of the groups he wants to import for. This restriction does not hold when importing the the Command Line Interface (CLI) ([link to CLI import g.doc](#)).

1.3 Import data using the Command Line Interface (CLI)

1.3.1 Description

This chapter will show how to import data for another user, using Command Line Interface (CLI).

The import can be done by any user as long as they import the data for themselves.

In case of the import for others, the user importing the data needs to have some admin (or restricted-admin) privileges. More information about restricted privileges can be found at [restricted-admins](#).

In the example workflow below, a user with restricted administrator privileges is used with login name `importer1`. This could be in real life e.g. a facility manager.

We will show:

- How to import data using the CLI for myself and for others into a remote OMERO.server
- How to import data using the CLI “in-place”, which means not copying the imported data into OMERO. Instead, OMERO will point to the original location of “in-place” imported files, thus preventing data duplication.
- How to deal with imports of large amounts of data in CLI, using the `-bulk` option and helper `csv` and `yml` files which define what is to be imported and how.

1.3.2 Resources

- Documentation:
 - <https://docs.openmicroscopy.org/latest/omero/users/cli/installation.html>
 - <https://docs.openmicroscopy.org/omero/latest/users/cli/index.html>
 - <https://docs.openmicroscopy.org/omero/latest/users/cli/import-target.html>
 - <https://docs.openmicroscopy.org/omero/latest/sysadmins/in-place-import.html>

- <https://docs.openmicroscopy.org/omero/latest/users/cli/import-bulk.html>
- Data: example images from
 - <https://downloads.openmicroscopy.org/images/DV/will/FRAP/>
- Bash script for performing in-place imports:
 - https://github.com/ome/training-scripts/blob/master/maintenance/scripts/in_place_import_as.sh
- Example files for bulk import
 - `bulk.yml`

1.3.3 Setup

CLI Importer installation

Client libraries from the OMERO.server have to be installed on the client to import images using CLI. Please read the [installation instructions](#).

Note: When importing for another user using the CLI, the `importer1` does not have to be a member of the target group.

1.3.4 Step-by-step

1. If you did not do so already, open a terminal window on your local machine and activate the conda environment where your `omero-py` is installed (see the Setup section above):

```
$ conda activate myenv
```

2. Set the `OMERODIR` variable to point to the downloaded and unzipped OMERO server dir (see the Setup section above):

```
$ export OMERODIR=/path/to/OMERO.server-x.x.x-ice36-bxx
```

3. Login to the OMERO.server you wish to import to. This can be a remote OMERO.server. In the below example we log in as the user `importer1`, who will import for themselves:

```
$ omero -s your-server-address -u importer1 login
```

4. Go to the directory where there are some images you wish to import:

```
$ cd /path/to/images/directory/
```

5. Create an OMERO Dataset to import to:

```
$ DID=$(omero obj new Dataset name=import_for_myself)
```

6. Import the images as `importer1` into the newly created Dataset. You can import a single image as in the example below, or a whole directory of images:

```
$ omero import -d $DID <your-image-name>
```

7. Log in to OMERO.web as `importer1` and verify the imported image in the newly created dataset `import_for_myself`.
8. Import an image for another user, for example `user-1`. For that, your `importer1` user logs in as `user-1`:

```
$ omero --sudo importer1 -u user-1 login
```

9. Create a Dataset `import_for_user_one` as `user-1`:

```
$ DID=$(omero obj new Dataset name=import_for_user_one)
```

10. Import the data in the newly created Dataset:

```
$ omero import -d $DID <path-to-image-or-directory-with-images>
```

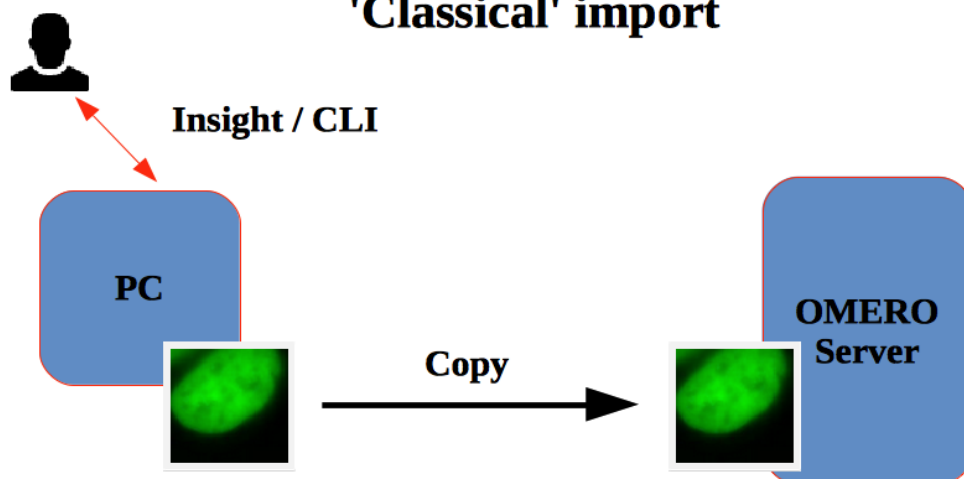
11. Check that the image(s) are successfully imported and that the image(s) and the containing Dataset both belong to `user-1` (not `importer1`).

1.4 In-place Import using the CLI

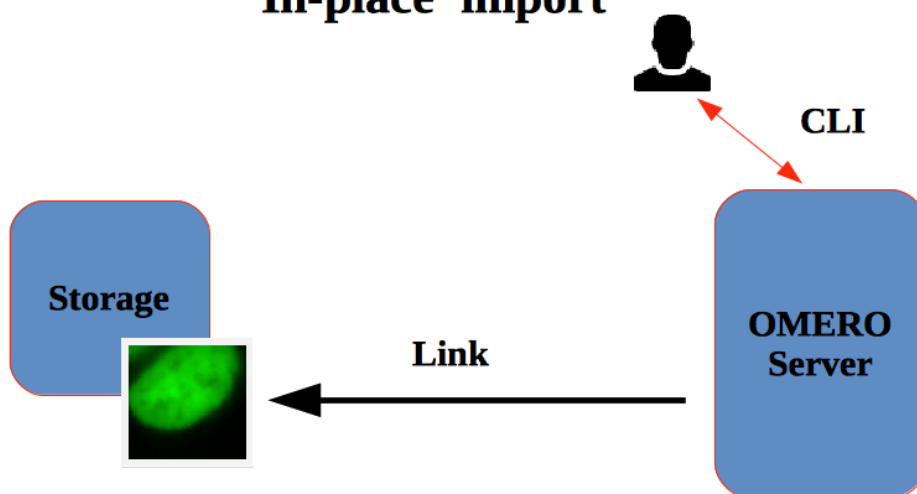
Instead of being copied into OMERO's managed repository, the image files stay at their original place and are just linked into the repository.

It is only available for the CLI importer, using the argument `--transfer=ln_s`.

'Classical' import



'In-place' import



Advantages:

- All in-place import scenarios provide non-copying benefit. Data that is too large to exist in multiple places, or which is accessed too frequently in its original form to be renamed, remains where it was originally acquired.

Limitations:

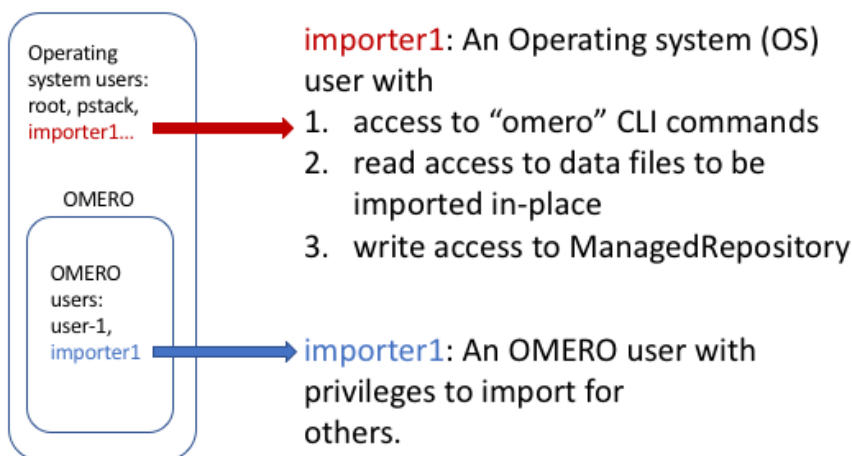
- Only available on the OMERO server system itself.
- Do not edit or move the files after an in-place import. OMERO may no longer be able to access them if you do.

Important:

A user performing an in-place import MUST have:

- a regular OMERO account
- an OS-account with ability to run omero commands on server machine
- read access to the location of the data
- write access to the ManagedRepository or one of its subdirectories. Please check the [ManagedRepository](#) documentation.

Unix machine with installed OMERO.server



1.4.1 Step-by-step:

1. Connect to the machine on which the OMERO.server is running as OS user `importer1` using `ssh`.
2. The aim is to import an image from `/OMERO/in-place-import/FRAP`:

```
$ ls /OMERO/in-place-import/FRAP
```

3. Activate the virtual environment where `omero-py` is installed or add it to `PATH`. In the example below, the path to the OMERO.server is `/opt/omero/server`:

```
$ export PATH=/opt/omero/server/venv3/bin:$PATH
```

4. Point `OMERODIR` to the location where the OMERO server is installed e.g.:

```
$ export OMERODIR=/opt/omero/server/OMERO.server
```


5. Import now data for another user, this time a large image where the advantage of not copying the image file onto the server is most visible. The `importer1` user logs in as `user-1`:



```
$ omero --sudo importer1 -u user-1 login
```

6. Create a Dataset `import_for_user_one`:

```
$ DID=$(omero obj new Dataset name=import_for_user_one)
```

- ‘In place’ import a large SVS file into the `import_for_user_one` dataset:

```
$ omero import -d $DID --transfer=ln_s /OMERO/in-place-import/svs/77917.svs
```

- Check that the image is successfully imported.
- Click on the paths icon  to show the difference between the normal and in-place (`ln_s`) imported images. Validate that In-place import is indicated .
- Note: The script `in_place_import_as.sh` shows how to perform the in-place import steps described above in one single command.

1.5 Bulk Import using the CLI

In this example, we show how to combine several import strategies using a configuration file. This is a strategy heavily used to import data to IDR.

We import two folders named *siRNA-HeLa* and *condensation*.

Note: Connecting over SSH is necessary only if you intend to import in-place. If you do not wish to perform the bulk import in “in-place” manner, you can connect to the server remotely using locally installed `OMERO.cli` and adjust the `bulk.yml` file by commenting out the `transfer...` line, then follow the steps as described below.

1. Open a terminal and connect to the server (for example as `importer1`) over SSH. Alternatively, use your local terminal with installed `OMERO.cli` if not importing “in-place”.
2. Description of the files used to set up the import (see `bulk.yml`, `import-paths.csv` and [import-bulk.html#bulk-imports](#) for further details).
 - `import-paths.csv`: (.csv, comma-separated values) this file has at least two columns. In this case the columns are separated by commas. The first column is the name of the target Dataset and the second one is the path to the folder to import. We will import two folders (the `import-paths.csv` has two rows).

Example csv (note the comma between the “HeLa” and “/OMERO...”):

```
*Dataset:name:Experiment1-HeLa,/OMERO/in-place-import/
siRNAi-HeLa*

*Dataset:name:Experiment2-condensation,/OMERO/in-place-import/
condensation*
```

- `bulk.yml`: this file defines the various import options: transfer option, checksum algorithm, format of the .csv file, etc. Note that setting the `dry_run` option to true allows to first run an import in `dry_run` mode and copy the output to an external file. This is useful when running an import in parallel. Comment out the `transfer "ln_s"` if not importing “in-place”.

Example `bulk.yml`:

```
*continue: "true"*

*transfer: "ln_s"*

*# exclude: "clientpath"*

*checksum_algorithm: "File-Size-64"*

*logprefix: "logs"*

*output: "yaml"*

*path: "import-paths.csv"*

*columns:*
  - *target*
  - *path*
```

3. Activate the virtual environment where `omero-py` is installed or add it to `PATH`. In the example below, the path to the `OMERO.server` is `/opt/omero/server`:

```
$ export PATH=/opt/omero/server/venv3/bin:$PATH
```

4. Point `OMERODIR` to the location where the `OMERO` server is installed e.g.:

```
$ export OMERODIR=/opt/omero/server/OMERO.server
```

5. Find the place where the `bulk.yml` file is located, for example `/OMERO/in-place-import`:

```
$ cd /OMERO/in-place-import
```

6. The `importer1` (Facility Manager with ability to import for others) `OMERO` user logs in as `user-1`:

```
$ omero --sudo importer1 -u user-1 login
```


7. Import the data using the `--bulk` command:


```
$ omero import --bulk bulk.yml
```

8. Go to the webclient during the import process to show the newly created dataset. The new datasets in `OMERO` are named `Experiment1-HeLa` and `Experiment2-condensation`. This was specified in the first column of the `import-paths.csv` file.

9. Select an image.

10. In the right-hand panel, select the `General` tab to validate:

- Click on  to show the import details.

- Validate that In-place import is indicated  in case you imported “in-place”.

Advantages:

- Large amount of data imported using one import command.

- Heterogeneous data for multiple users can be imported using bulk import in combination with bash scripting, e.g. `in_place_import_as.sh`
- Reproducible import.

Limitations:

- Preparation of the `.csv` or `.tsv` file.

1.6 Combine the CLI imports with post-import steps

The following example shows how to do the import on CLI and follow-up operations like rendering and metadata import in one step.

In many cases, the rendering and metadata import is best done separately, as the visual checking of the imported images might be crucial for further rendering and metadata import, see *Change image rendering settings and channel names using the Command Line Interface (CLI)* and *Import metadata using the Command Line Interface (CLI)* for details on this.

Further, the images you are importing might need a range of different rendering settings, not just one set of settings for all of them. Also for this case, the step-by-step approach, first importing the images, only then deciding on the rendering strategy and preparing the `renderingdef.yml` files, is preferable.

Nevertheless, there are cases which do not need visual checks and use a single rendering for all images, for which a streamlined sequence of commands is offered below which will perform all three steps (import, rendering and metadata import) in one single session on the CLI.

1.6.1 Resources

Additionally to the *Resources mentioned in the import-cli section* and in the *Setup* you will also need the rendering and metadata plugins as mentioned in *Change image rendering settings and channel names using the Command Line Interface (CLI)* and *Import metadata using the Command Line Interface (CLI)*, and possibly the following files:

- <https://github.com/ome/training-scripts/blob/master/maintenance/preparation/renderingdef.yml>
- `simple-annotation.csv`
- `simple-annotation-bulkmap-config.yml`

1.6.2 Step-by-step

1. If you did not do so already, open a terminal window on your local machine and activate the conda environment where your `omero-py` is installed (see the *Setup*):

```
$ conda activate myenv
```

2. Set the `OMERODIR` variable to point to the downloaded and unzipped OMERO server dir (see the *Setup*):

```
$ export OMERODIR=/path/to/OMERO.server-x.x.x-ice36-bxx
```

3. Prepare an `renderingdef.yml` file, by either creating a new one or downloading <https://raw.githubusercontent.com/ome/training-scripts/master/maintenance/preparation/renderingdef.yml>.
4. Prepare an `annotation.csv` file, by creating a new file or downloading the provided example file. In the example below, we use the file `simple-annotation.csv`. The Dataset names in this CSV file must match the

Dataset names in OMERO as created in the *DID* variable definition line in the command below. The Image names in the CSV file must match the file names in your imported folder.

5. Prepare a *bulkmap-config.yml* file. In the example below, we use the file *simple-annotation-bulkmap-config.yml*.
6. Log in to the OMERO.server you wish to import to. This can be a remote server if you do not wish to import *in-place*.
7. Import, render and annotate in a single command sequence below:

```
$ PID=$(omero obj new Project name='Project_import_concatenate')
$ DID=$(omero obj new Dataset name='siRNAi-HeLa')
$ omero obj new ProjectDatasetLink parent=$PID child=$DID
$ omero import -d $DID /path/to/data/folder/or/image/siRNAi-HeLa --file import.
↪out
$ omero render set $DID renderingdef.yml
$ omero metadata populate --report --batch 1000 --file /path/to/downloaded/
↪simple-annotation.csv $PID
$ omero metadata populate --context bulkmap --cfg simple-annotation-bulkmap-
↪config.yml --batch 100 $PID
```

8. Log in to OMERO.web and check that the images are imported, have the expected rendering settings and also the annotations in the form of Key-Value pairs on each imported image.

For more information about CLI import options, go to [import.html](#).

1.7 Import data using OMERO.dropbox

1.7.1 Description

OMERO.dropbox allows to import files into OMERO automatically, by means of offline import from a watched directory. Typically, each user has a folder into which they “drop” their images as these are being acquired. The folder can be directly on the machine where OMERO is installed, or, more commonly, OMERO can watch directories mounted on the machine where OMERO is installed.

Alternatively, the files to import can be copied into the folders watched by OMERO.dropbox by cron jobs <https://en.wikipedia.org/wiki/Cron> or systems developed by the community users such as <https://github.com/imcf/auto-tx>, which automatically harvest the files from acquisition computers and transfer them onto shared network drives.

We will show

- How to set up OMERO.dropbox on your OMERO.server.
- How to set up the directories/folders into which the users or the automatic systems described above will copy the files to import.
- How to import several files for two different users using OMERO.dropbox. The image files are manually copied into the prepared folders watched by OMERO.dropbox.

1.7.2 Resources

- Documentation:
 - <https://docs.openmicroscopy.org/latest/omero/sysadmins/dropbox.html>
 - <https://docs.openmicroscopy.org/omero/latest/users/cli/import-target.html>

- Data: example images from <https://downloads.openmicroscopy.org/images/DV/alexia/cajal-bodies/>

1.7.3 Setup

This example setup will help you to understand OMERO.dropbox functionality. Later, you can choose the setup you like for your OMERO.server studying detailed instructions at <https://docs.openmicroscopy.org/latest/omero/sysadmins/dropbox.html#advanced-use>

Below are the installation instructions.

- First connect to the OMERO.server over SSH using your administrator account.
- In the OMERO.server directory set the following lines to enable and configure Dropbox:
 - `$ bin/omero config set omero.fs.watchDir "/home/DropBox"`
 - `$ bin/omero config set omero.fs.importArgs "-T \\\"regex:^(.*/(?:<Container1>.*)?\\\""`

Note: The last line makes sure the created Dataset into which the images will be imported will have the same name as the deepest folder in the DropBox folder for that particular user. Every image in any Experiment-1 directory will end up in the same Dataset for that user, however the superstructure of folders have been.

To set up different import options, go to <https://docs.openmicroscopy.org/omero/latest/users/cli/import-target.html>.

For example, it is possible to create a Dataset with a fixed name for every user or to create a Dataset whose name is picked from the first subfolder under images folder.

- On the OMERO.server machine, create a folder called DropBox under /home.
- `$ cd /home`
- `$ mkdir DropBox`
- Create under the /home/DropBox directory two subdirectories for two users e.g. user-1 and user-2. The names of these directories should match the login names of those users in OMERO.

Note: the omero system user must be able to read from those directories. Also, the users or the system which will drop files into those directories must have write permissions there.

- `$ cd /home/DropBox`
- `$ mkdir user-1`
- `$ mkdir user-2`

1.7.4 Step-by-step

1. Open a browser window.
2. Enter the URL provided.
3. Login as user-1 or any user with the right to see user-1's data.
4. Leave the browser window open.
5. On your computer, open a new terminal window.
6. Connect over SSH to the machine where the OMERO.server is running.
7. Open the logfile DropBox.log under var/log/ e.g.

- `$ tail -f /path/to/OMERO.server/var/log/DropBox.log`
8. Open a new terminal window.
 9. Connect over SSH to the machine where the OMERO.server is running.
 10. In any folder on that machine, create a directory Experiment-1 into which you copy an image. Drop the Experiment-1 directory into the user-1 folder you created during the setup above:
 - `$ cd /path/to/other/dir`
 - `$ mkdir Experiment-1`
 - `$ scp 090829_5_HeLa_siCTL_coilin_ATUB01_05_R3D_D3D.dv ./Experiment-1`
 - `$ cd /home/DropBox`
 - `$ scp -r /path/to/other/dir/Experiment-1 ./user-1 #copy the whole directory "Experiment-1" into the directory watched by OMERO`
 11. The OMERO.dropbox will intentionally wait for 60 seconds between registration of the new drop into the folder and the actual import, in anticipation of further file drops of files into the DropBox folder.
 12. After you have copied the directory into the user-1 folder, you should see in the DropBox.log lines like as follows.

```
2019-08-12 17:09:23,644 INFO [ fsclient.fsDropBoxMonitorClient]
(Thread-3 ) New entry
/home/DropBox/user-1/Experiment-1/090829_5_HeLa_siCTL_coilin_ATUB01_05_R3D_D3D.dv
contains 1 file(s). Files=1 Timers=1

2019-08-12 17:10:23,644 INFO [ fsclient.fsDropBoxMonitorClient]
(Thread-17 ) Removed key
/home/DropBox/user-1/Experiment-1/090829_5_HeLa_siCTL_coilin_ATUB01_05_R3D_D3D.dv

2019-08-12 17:10:23,666 INFO [ fsclient.fsDropBoxMonitorClient]
(Thread-17 ) Importing
/home/DropBox/user-1/Experiment-1/090829_5_HeLa_siCTL_coilin_ATUB01_05_R3D_D3D.dv
(session=2155e6d0-445a-496b-a6bc-8afeb93ac58d)

2019-08-12 17:10:23,955 INFO [ omero.util.Resources] (Thread-18 )
Starting

2019-08-12 17:10:23,961 WARNI [ stderr] (Thread-17 ) Joined session
for user-2@localhost:4064. Idle timeout: 10 min. Current group: Lab1

2019-08-12 17:10:31,989 INFO [ fsclient.fsDropBoxMonitorClient]
(Thread-17 ) Import of
/home/DropBox/user-1/Experiment-1/090829_5_HeLa_siCTL_coilin_ATUB01_05_R3D_D3D.dv
completed (session=2155e6d0-445a-496b-a6bc-8afeb93ac58d)

2019-08-12 17:10:32,001 INFO [ omero.util.Resources] (Thread-18 )
Halted
```

1. Go back to OMERO.web. Refresh the tree.
2. Observe that a new Dataset was created, with the name Experiment-1. The image is imported into that Dataset.
3. The image is always imported into the default group of the user.
4. Repeat the workflow for user-2. First, go to your browser, logout and login again as user-2.
5. Connect over SSH to the machine where the OMERO.server is running if required.

6. Create again a folder, Experiment-2.
7. Copy an image into it
8. Copy the whole Experiment-2 folder under /home/DropBox/user-2.
9. Go back to the browser, refresh and verify that you can see a Dataset Experiment-2 under user-2's data with the image inside.
10. Note: Even if user-2's folder in the previous workflow uses the same name for their dataset as user-1 (Experiment-1), the data of user-2 would not be imported into user-1's Dataset. Instead, a new Dataset Experiment-1 would be created under user-2's data, belonging to user-2, into which the image would be imported.

Change image rendering settings and channel names using the Command Line Interface (CLI)

2.1 Description

This chapter will show how to change rendering settings on images using the Command Line Interface (CLI).

This action is typically done after a successful import of images.

We will show:

- How to change rendering settings of large amount of images on the CLI in a repeatable manner.

2.2 Resources

- Documentation:
 - <https://docs.openmicroscopy.org/latest/omero/users/cli/installation.html>
 - <https://docs.openmicroscopy.org/omero/latest/users/cli/index.html>
- Data: example images from
 - IDR data `idr0021`.
- Rendering plugin for OMERO
 - <https://pypi.org/project/omero-cli-render/>
- Rendering yaml files defining the various rendering parameters, such as the channel color, channel name and minimum and maximum values.
 - <https://github.com/ome/training-scripts/blob/master/maintenance/preparation/renderingdef.yml>
 - <https://github.com/ome/training-scripts/blob/master/maintenance/preparation/renderingdef2.yml>
- Rendering mapping file has two columns, in the left-hand one there are the images to be rendered and in the right-hand side column points to the appropriate renderingdef.yml for that image.

- <https://github.com/ome/training-scripts/blob/master/maintenance/preparation/renderingMapping.tsv>
- Shellscript for batch modification of rendering settings of images
 - https://github.com/ome/training-scripts/blob/master/maintenance/scripts/apply_rnd_settings_as.sh

2.3 Setup

Rendering plugin installation

- Go to the environment where you installed your OMERO.cli as specified under - <https://docs.openmicroscopy.org/latest/omero/users/cli/installation.html>.
- Activate the virtual environment.
- Run:

```
$ pip install omero-cli-render
```

2.4 Step-by-step

1. On your local machine, open a terminal
2. Activate the virtual environment where `omero-py` is installed or add it to `PATH` e.g.:

```
$ export PATH=/opt/omero/server/venv3/bin:$PATH
```

3. The variable `$ID` below is the ID of the selected Dataset. To change the rendering of images in one Dataset, run:

```
$ omero render set Dataset:$ID local_path/to/renderingdef.yml
```

4. Verify the change in the browser.
5. To change the rendering of images in two Datasets, run:

```
$ omero render set Dataset:$ID1 Dataset:$ID2 renderingdef2.yml
```

6. Modify the rendering settings in batch using a shellscript such as `apply_rnd_settings_as.sh` which uses HQL to find the Images IDs in OMERO and deliver them to the `omero-cli-render` plugin. The script reads the Datasets in which the Images are located in OMERO are listed from a `renderingMapping.tsv` file, such as `renderingMapping.tsv`. Go to the folder where the script is located. Then run the bash script with the default parameters:

```
$ sh apply_rnd_settings.sh
```

The script could be run by a facility manager on behalf of other users.

Import metadata using the Command Line Interface (CLI)

3.1 Description

This chapter will show how to import metadata starting from a local CSV file and ending with OMERO.tables on images or Key-Value pairs on images using the Command Line Interface (CLI). For a more user-friendly way of uploading metadata using graphical interface see the *Import metadata using the Populate Metadata script in OMERO.web* chapter.

This action is typically done after a successful import of images.

We will show:

- How to import metadata from local CSV file in a bulk manner and turn them into OMERO.tables on images using CLI
- How to turn the OMERO.tables on images into Key-Value pairs on images in bulk manner using CLI
- How to import metadata from local CSV file and use a server-side script in OMERO to turn these into OMERO.tables on images
- How to construct a simple file to turn the metadata stored in OMERO.tables into Key-Value pairs on images using CLI

3.2 Resources

- Documentation:
 - [CLI installation](#)
 - [CLI](#)
- Data: example images from
 - [IDR data idr0021](#)
 - [siRNAi-HeLa dataset](#)

- Metadata plugin for OMERO
 - <https://pypi.org/project/omero-metadata/>
- Bulkmap config yaml files defining the various Key-Value pairs parameters, such as the groups and other parameters.
 - `idr0021-experimentA-bulkmap-config.yml`
 - `simple-annotation-bulkmap-config.yml`
- Annotation CSV files define the content of OMERO.tables for each image.
 - `idr0021-experimentA-annotation.csv`
 - `simple-annotation.csv`
 - `four-images.csv`

3.3 Setup

Metadata plugin installation

- Go to the environment where you installed your OMERO.cli as specified under - [CLI installation](#).
- Activate the virtual environment.
- Run:

```
$ pip install omero-metadata
```

3.4 Step-by-step

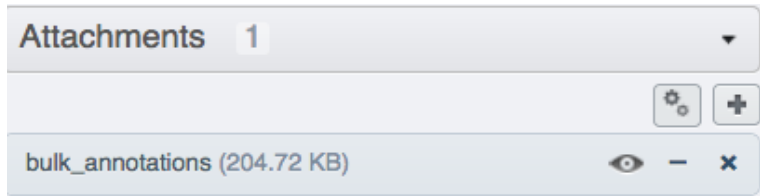
1. On your local machine, open a terminal
2. If you did not do so already, activate the virtual environment where `omero-py` is installed or add it to PATH e.g.:


```
$ export PATH=/opt/omero/server/venv3/bin:$PATH
```

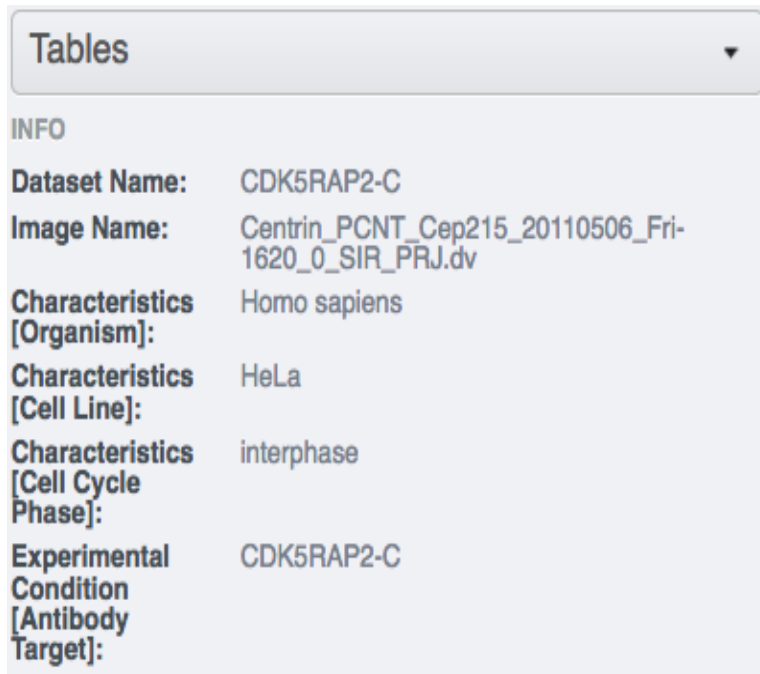
3. Download the CSV from `idr0021-experimentA-annotation.csv` if you have access to the `idr0021` data in your OMERO.server. Alternatively, download `simple-annotation.csv`, which will allow you to work with the `siRNAi-HeLa` dataset.
4. The variable `$ID` below is the ID of the Project, in this example case it is the Project containing the `idr0021` study. If you are working with the `siRNAi-HeLa` data, replace in the following example the “Project” with a “Dataset” and the `idr0021-experimentA-annotation.csv` with `simple-annotation.csv`. To add annotations from a local CSV file to the images in the said Project or Dataset in the form of OMERO.tables, run:

```
$ omero metadata populate --report --batch 1000 --file local/path/to/idr0021-  
↪experimentA-annotation.csv Project:$ID
```

5. Open your browser and login to the OMERO.web. Navigate to the Project or Dataset you just worked with, expand the “Attachments” harmonica in the right-hand pane and verify that a new attachment is on that Project named `bulk_annotations`.



6. You can inspect its content by clicking on the “eye” icon  inside the annotation.
7. Select an image inside the Project/Dataset and expand the “Tables” harmonica in the right-hand pane. These tables contain the appropriate line from the `bulk_annotations` attachment you just created for that particular image.



8. Go back to your terminal. Download the `idr0021-experimentA-bulkmap-config.yml` file . Alternatively, in case you are working with the siRNAi-HeLa Dataset, download `simple-annotation-bulkmap-config.yml`.
9. If you are working with the IDR data, open the downloaded `idr0021-experimentA-bulkmap-config.yml` file in a text editor and delete the `Advanced options...` section. Save the file and run:

```
$ omero metadata populate --context bulkmap --cfg local/path/to/idr0021-
↳experimentA-bulkmap-config.yml --batch 100 Project:$ID
```

10. If you work with the siRNAi-HeLa data, open the downloaded `simple-annotation-bulkmap-config.yml` and study the comments in the file itself, which will give you hints about how to manipulate the file to fit your particular needs with respect to the resulting Key-Value pairs layout. Make your changes (no need to change anything if you do not want), save the file locally and run:

```
$ omero metadata populate --context bulkmap --cfg local/path/to/simple-annotation-
↳bulkmap-config.yml --batch 100 Dataset:$ID
```

11. Go to your browser and in OMERO.web, select the images in the Project or Dataset you targeted and verify that they have now new Key-Value pairs displayed in the right-hand pane.

Key-Value Pairs
7

openmicroscopy.org/mapr/cell_line
+
📄
🗑️
✕

Added by: Maurice Wilkins

Cell Line	HeLa
-----------	------

Gene

Added by: Maurice Wilkins

Gene Identifier	ENST00000394818.8	<input type="checkbox"/>
Gene Symbol	INCENP	
Gene Symbol Synonyms	FLJ31633	

openmicroscopy.org/mapr/organism

Added by: Maurice Wilkins

Organism	Homo sapiens
----------	--------------

Import metadata using the Populate Metadata script in OMERO.web

4.1 Description

This chapter will show how to import metadata starting from a local CSV file and ending with OMERO.tables on Images or wells using the [server-side script](#) Populate Metadata in OMERO.web. See also the workflow described in [Import metadata using the Command Line Interface \(CLI\)](#) which will give you more possibilities, such as Key-Value Pairs creation, but which is using Command Line Interface (CLI). The workflow described here is using only graphical user interface elements.

This action is typically done after a successful import of Images.

We will show:

- **How to import metadata from local CSV file in a bulk manner and turn them into OMERO.tables on**
 - *Images contained in Projects/Datasets*
 - *Wells contained in Screens/Plates*
- How to *create or adjust a local CSV file containing metadata* for creation of OMERO.tables containing numbers and text.

4.2 Resources

- Annotation CSV files define the content of OMERO.tables for each image or each well.
 - `four-images.csv`
 - `simple-screen.csv`
- `omero-metadata` plugin (necessary for having full set of features in Populate Metadata script).
 - <https://pypi.org/project/omero-metadata/>

4.3 Setup

omero-metadata plugin installation

Note: For the best experience, the `omero-metadata` plugin should be installed on your OMERO.server. The `Populate Metadata` script tries to reuse the code of the `omero-metadata` plugin. If the plugin is not found on the server, the `Populate Metadata` script falls back on a deprecated code, with limited set of features. The `omero-metadata` plugin installation is typically done by the administrator of the OMERO.server.

- In your OMERO.server environment, go to the environment where you installed your OMERO.cli as specified under - [CLI installation](#).
- Activate the virtual environment.
- Run:

```
$ pip install omero-metadata
```

Populate Metadata script

No explicit installation necessary, shipped with the OMERO.server.

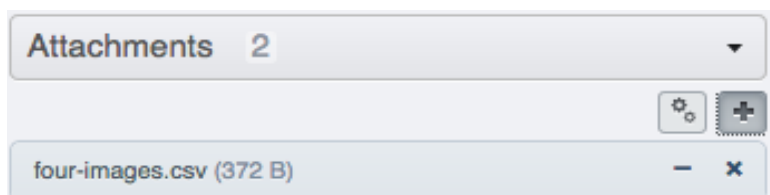
4.4 Step-by-step

4.4.1 Project/Dataset/Image

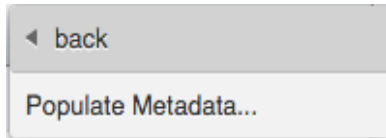
1. Log in to OMERO.web, create a new Dataset and copy into it four Images, preferably Images which have no OMERO.tables on them. Note the name of the Images you are copying in.



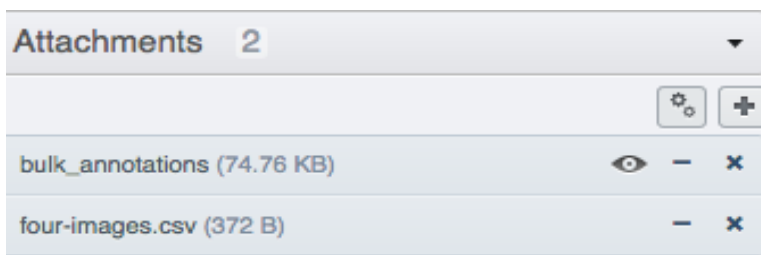
2. Download `four-images.csv`. Open the CSV file in Excel and edit the name of the Images in the first column to match the names of the Images you copied into your Dataset in the previous step. Also, edit the name of the Dataset in the second column to match the name of your Dataset in OMERO.web. Save the file locally as CSV.
3. (Optional) In your OMERO.web, upload the CSV file you just saved and attach it onto the Dataset you created previously. Alternatively, you can skip this step, and point the `Populate Metadata` script to the local CSV, as explained below.



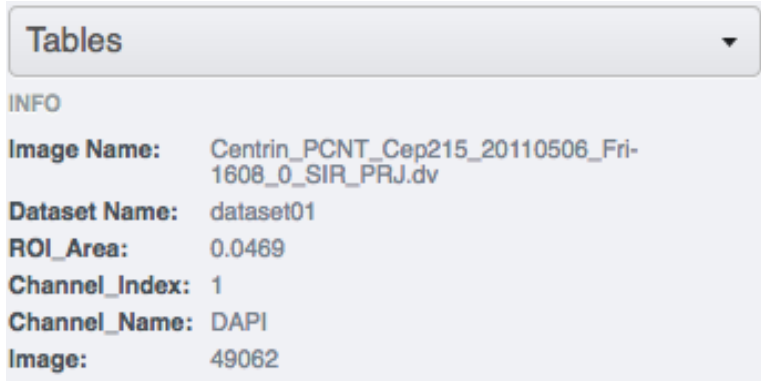
4. Select the Dataset you created. Find the script icon  above the central pane, expand it and find the `Import scripts` section. In there, select the `Populate metadata` script which will launch the script dialog.



5. If you did not attach the CSV to the Dataset, you can now click on the `Browse` button and select the CSV from your local machine.
6. Click `OK` to run the script, and wait for it to show as complete in the `Activities` panel in the top-right corner above the central pane.
7. Click again onto the Dataset in the left-hand pane to refresh and observe that there is a new Attachment in the right hand pane under “Attachments” harmonica, named `bulk_annotations`.

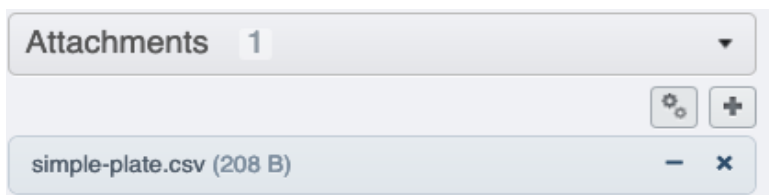


8. Click on single Images inside the Dataset and observe that in the “Tables” harmonica in the right-hand pane there are new values coming originally from your edited CSV.

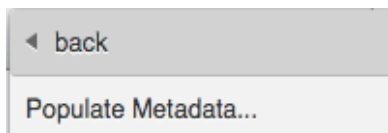


4.4.2 Screen/Plate/Well

- Find a Plate inside a Screen in OMERO.web which has no OMERO.tables on its Wells.
- Download `simple-screen.csv`. Open the CSV file in Excel and edit the name of the wells in the first column to match the names of the wells in your Plate from the previous step. Also, edit the name of the Plate in the second column to match the name of your Plate in OMERO.web. Save the file locally as CSV.
- (Optional) In your OMERO.web, upload the CSV file you just saved and attach it onto the Screen containing the Plate you created previously. Alternatively, you can skip this step, and point the `Populate Metadata` script to the local CSV, as explained below.



4. Select the Screen you identified above. Find the script icon  above the central pane, expand it and find the Import scripts section. In there, select the Populate metadata script which will launch the script dialog.



5. If you did not attach the CSV to the Screen, you can now click on the Browse button and select the CSV from your local machine.
6. Click OK to run the script, and wait for it to show as complete in the Activities panel in the top-right corner above the central pane.
7. Click again onto the Screen in the left-hand pane to refresh and observe that there is a new Attachment in the right hand pane under Attachments harmonica, named bulk_annotations.
8. Click on single Wells inside the Plate under the Screen and observe that in the Tables harmonica in the right-hand pane there are new values coming originally from your edited CSV.

4.4.3 Create a metadata CSV

1. Download the four-images.csv (for Images in Projects/Datasets) or simple-screen.csv (for Wells in Screens/Plates) as templates to create your own CSV.
2. Open the downloaded CSV file in Microsoft EXcel, but do not use Import command in Excel, instead, either double-click on the file or use the Open command in Excel. Populate the values in the CSV using Microsoft Excel with your own numbers or text, possibly expanding the number of rows or columns as appropriate.
3. Replace the # header . . . column types inside the templates with your own column types according to the content of your CSV: Follow the Note below for guidelines. Save the file as CSV in Microsoft Excel.

Note: The # header row is optional. If # header is not used, all column types are treated as String (i.e. text, not numbers) in OMERO. The header abbreviations have following meaning:

d: DoubleColumn, for floating point numbers

l: LongColumn, for integer numbers

s: StringColumn, for text

b: BoolColumn, for true/false

plate, well, image, dataset, roi: to specify objects

If the target is a Project, the CSV file needs to specify Dataset Name and Image Name. If the target is a Dataset instead of a Project, the Dataset Name column is not needed.

If the target is a Screen, the CSV file needs to specify Plate name and Well. If a # header is specified, column types must be well and plate. If the target is a Plate, the CSV file **must not** specify a Plate column, but it must specify the Well column.

Column names should not contain spaces if you want to be able to query by these columns.

CHAPTER 5

Contribute

Changes to the documentation or the materials should be made directly in the [omero-guide-upload repository](#).