## I About the OMERO Platform

1 Introduction 3

2 OMERO clients 4
   2.1 OMERO client overview 4
   2.2 Command Line Interface as an OMERO client 6

3 Additional resources 24
   3.1 Community support 24
   3.2 What’s new for OMERO 5.2 for users 25
   3.3 OMERO version history 25

4 Quickstart server access 42
   4.1 OMERO demo server 42
   4.2 OMERO virtual appliance 45

## II System Administrator Documentation

5 Server Background 61
   5.1 What’s new for OMERO 5.2 for sysadmins 61
   5.2 Server overview 61
   5.3 System requirements 62
   5.4 Version requirements 64
   5.5 Example production server set-ups 76
   5.6 Known limitations 78
   5.7 Groups and permissions system 80

6 Server Installation and Maintenance 85
   6.1 Server installation on UNIX (and UNIX-like platforms) 85
   6.2 Server installation on Windows 145
   6.3 Command Line Interface as an OMERO admin tool 173
   6.4 Upgrading and maintenance 184
   6.5 Installing additional features 196
   6.6 Troubleshooting OMERO 208

7 Optimizing Server Configuration 215
   7.1 Server security and firewalls 215
   7.2 LDAP authentication 218
   7.3 Performance and monitoring 222
   7.4 Search and indexing configuration 226
   7.5 FS configuration options 229
   7.6 Grid configuration 231
   7.7 Setting the OMERO_HOME environment variable 234
   7.8 Configuration properties glossary 235
   7.9 Syslog configuration 260

8 Optimizing OMERO as a Data Repository 263
8.1 Customize OMERO clients .................................................. 263
8.2 Public data in the repository ................................................. 267

9 Data Import and Storage ..................................................... 269
9.1 OMERO.dropbox ............................................................. 269
9.2 In-place import ............................................................... 274
9.3 Advanced import scenarios ................................................. 281

III Developer Documentation .................................................. 286

10 Introduction to OMERO ..................................................... 288
10.1 What’s new for OMERO 5.2 for developers ........................... 288
10.2 Installing OMERO from source .......................................... 289
10.3 Build System ............................................................... 290
10.4 Working with OMERO ..................................................... 294
10.5 Running and writing tests ................................................. 300

11 Using the OMERO API ...................................................... 309
11.1 OMERO Python language bindings .................................... 309
11.2 Blitz Gateway documentation .......................................... 329
11.3 Command Line Interface as an OMERO development tool ....... 333
11.4 OMERO Java language bindings ....................................... 335
11.5 OMERO MATLAB language bindings ................................. 348
11.6 OMERO C++ language bindings ...................................... 363

12 Analysis ............................................................................. 370
12.1 Local analysis ............................................................... 370
12.2 Storing external data in OMERO ....................................... 370
12.3 OMERO.tables .............................................................. 371

13 Scripts - plugins for OMERO .............................................. 377
13.1 Introduction to OMERO.scripts ........................................ 377
13.2 OMERO.scripts user guide .............................................. 380
13.3 Guidelines for writing OMERO.scripts ............................... 386
13.4 MATLAB and Python ..................................................... 389
13.5 OMERO.scripts advanced topics .................................... 391

14 Web .................................................................................... 395
14.1 OMERO.web framework ................................................ 395
14.2 OMERO.web deployment for developers ............................ 398
14.3 Creating an app ............................................................. 399
14.4 Webclient Plugins .......................................................... 404
14.5 Editing OMERO.web ..................................................... 409
14.6 WebGateway ............................................................... 410
14.7 Embedding an OMERO.web viewport in a web page .......... 414
14.8 Writing OMERO.web views ............................................ 415
14.9 Writing page templates in OMERO.web ............................ 419
14.10 Cross Site Request Forgery protection ............................... 422
14.11 The OMERO.web client application ................................. 423

15 Insight .............................................................................. 425
15.1 Architecture ................................................................. 425
15.2 Configuration ............................................................... 427
15.3 Contributing to OMERO.insight ...................................... 432
15.4 Directory contents .......................................................... 432
15.5 Event bus ...................................................................... 433
15.6 Event ............................................................................. 434
15.7 How to build an agent ..................................................... 437
15.8 How to build an agent’s view .......................................... 440
15.9 Retrieve data from server ................................................ 442
15.10 Organization ............................................................... 446
## 15.11 Taskbar

447

## 16 More on API Usage

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.1 Developing OMERO clients</td>
<td>453</td>
</tr>
<tr>
<td>16.2 OMERO Application Programming Interface</td>
<td>489</td>
</tr>
<tr>
<td>16.3 OMERO admin interface</td>
<td>492</td>
</tr>
<tr>
<td>16.4 Deleting in OMERO</td>
<td>493</td>
</tr>
<tr>
<td>16.5 OMERO Import Library</td>
<td>495</td>
</tr>
<tr>
<td>16.6 TempFileManager</td>
<td>496</td>
</tr>
<tr>
<td>16.7 Exception handling</td>
<td>497</td>
</tr>
<tr>
<td>16.8 Omero logging</td>
<td>503</td>
</tr>
<tr>
<td>16.9 Using the new 5.1 graph requests</td>
<td>505</td>
</tr>
</tbody>
</table>

## 17 The OME Data Model

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.1 OME-Remote Objects</td>
<td>508</td>
</tr>
<tr>
<td>17.2 Structured annotations</td>
<td>518</td>
</tr>
<tr>
<td>17.3 Glossary of all OMERO Model Objects</td>
<td>521</td>
</tr>
<tr>
<td>17.4 Units</td>
<td>580</td>
</tr>
<tr>
<td>17.5 Key-value pairs</td>
<td>583</td>
</tr>
<tr>
<td>17.6 Available transformations</td>
<td>585</td>
</tr>
</tbody>
</table>

## 18 Searching

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.1 OMERO search</td>
<td>587</td>
</tr>
<tr>
<td>18.2 File parsers</td>
<td>590</td>
</tr>
<tr>
<td>18.3 Search bridges</td>
<td>591</td>
</tr>
</tbody>
</table>

## 19 Authentication and Security

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.1 Password Provider</td>
<td>594</td>
</tr>
<tr>
<td>19.2 LoginAttemptListener</td>
<td>595</td>
</tr>
<tr>
<td>19.3 LDAP plugin design</td>
<td>595</td>
</tr>
<tr>
<td>19.4 OMERO roles</td>
<td>596</td>
</tr>
<tr>
<td>19.5 OMERO security system</td>
<td>597</td>
</tr>
<tr>
<td>19.6 OMERO permissions history, querying and usage</td>
<td>601</td>
</tr>
</tbody>
</table>

## 20 OMERO.server in depth

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.1 OMERO.server overview</td>
<td>610</td>
</tr>
<tr>
<td>20.2Extending OMERO.server</td>
<td>611</td>
</tr>
<tr>
<td>20.3 OMERO.blitz</td>
<td>618</td>
</tr>
<tr>
<td>20.4 OMERO.fs</td>
<td>619</td>
</tr>
<tr>
<td>20.5 Import under OMERO.fs</td>
<td>619</td>
</tr>
<tr>
<td>20.6 OMERO.processor</td>
<td>623</td>
</tr>
<tr>
<td>20.7 OMERO.server image rendering</td>
<td>624</td>
</tr>
<tr>
<td>20.8 Clustering</td>
<td>624</td>
</tr>
<tr>
<td>20.9 Collection counts</td>
<td>625</td>
</tr>
<tr>
<td>20.10 How To create a service</td>
<td>626</td>
</tr>
<tr>
<td>20.11 OMERO sessions</td>
<td>627</td>
</tr>
<tr>
<td>20.12 Aspect-oriented programming</td>
<td>630</td>
</tr>
<tr>
<td>20.13 OmeroContext</td>
<td>631</td>
</tr>
<tr>
<td>20.14 OMERO events and provenance</td>
<td>633</td>
</tr>
<tr>
<td>20.15 Properties</td>
<td>633</td>
</tr>
<tr>
<td>20.16 Using server queries internally</td>
<td>634</td>
</tr>
<tr>
<td>20.17 OMERO throttling</td>
<td>636</td>
</tr>
<tr>
<td>20.18 OMERO rendering engine</td>
<td>637</td>
</tr>
<tr>
<td>20.19 Scaling Omero</td>
<td>637</td>
</tr>
<tr>
<td>20.20 SqlAction</td>
<td>639</td>
</tr>
<tr>
<td>20.21 Model graph operations</td>
<td>641</td>
</tr>
</tbody>
</table>

Index

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>648</td>
</tr>
</tbody>
</table>

Index

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>649</td>
</tr>
</tbody>
</table>
The OMERO 5.2.6 documentation is divided into three parts:

About the OMERO Platform introduces the user-facing client applications and how to get started, details the CLI client, and indicates where users can access further help and support.

System administrators wanting to install and manage an OMERO server can find instructions in the System Administrator Documentation.

Developers can find more specific and technical information about OMERO in the Developer Documentation.

Additional online resources can be found at:

- Downloads
- Features (movie tutorials coming soon)
- Security vulnerabilities
- OMERO API documentation - OmeroJava API, OmeroPy API, OmeroBlitz / Slice API
- User help website
- Script sharing service
- Partner projects to extend OMERO
- Demo server - now managed by the main OME team, providing the latest released version of OMERO for you to try out (barring a short lag time for upgrading).

OMERO version 5.2 uses the January 2015 release of the OME Data Model. The OMERO version history page details the development of OMERO functionality over time.

A summary of breaking changes and new features for version 5.2 can be found on the pages below:

- What’s new for OMERO users
- What’s new for OMERO sysadmins
- What’s new for OMERO developers

The source code is hosted on Github. To propose changes and fix errors, go to the documentation repository, fork it, edit the file contents and propose your file changes to the OME team using Pull Requests. Alternatively, click on “Edit on GitHub” in the menu.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.
Part I

About the OMERO Platform
OMEReObjects (OMERO) is a modern client-server software platform for visualizing, managing, and annotating scientific image data. OMERO lets you import and archive your images, annotate and tag them, record your experimental protocols, and export images in a number of formats. It also allows you to collaborate with colleagues anywhere in the world by creating user groups with different permission levels. OMERO consists of a Java server, several Java client applications, as well as Python and C++ bindings and a Django-based web application.

The OMERO clients are cross-platform. To run on your computer they require Java 1.7 or higher to be installed. This can easily be installed from [http://java.com/](http://java.com/) if it is not already included in your OS. The OMERO.insight client gets all of its information from a remote OMERO.server — the location of which is specified at login. Since this connection utilises a standard network connection, the client can be run anytime the user is connected to the internet.

This documentation is for the OMERO 5 Platform. This version is designed to improve our handling of complex multidimensional datasets. It allows you to upload your files in their original format, preserving file names and any nested directory structure in the server repository. For more technical information, please refer to the Developer Documentation. You can read about the development of OMERO in the OMERO version history.
2.1 OMERO clients overview

Most laboratories use a number of different imaging platforms and thus require tools to manage, visualize and analyze heterogeneous sets of image data recorded in a range of file formats. Ideally a single set of applications, running on a user’s laptop or workstation, could access all sets of data, and provide easy-to-use access to this data.

OMERO ships as a server application called OMERO.server and a series of client applications (known simply as clients): OMERO.web, OMERO.insight and OMERO.importer. All run on the major operating systems and provide image visualization, management, and annotation to users from remote locations. With a large number of OMERO.server installations worldwide, OMERO has been shown to be relatively easy to install and get running.

OMERO.insight and OMERO.importer are desktop applications written in Java and require Java 1.7 (or higher) to be installed on the user’s computer (this can easily be installed from http://java.com/ if it is not already included in your OS).

Our user assistance help website\(^1\) provides a series of workflow-based guides to performing common actions in the client applications, such as importing and viewing data, exporting images and using the measuring tool.

Our partners within the OME consortium are also producing new clients and modules for OMERO, integrating additional functionality, particularly for more complex image analysis. See the Partner Projects\(^2\) page for more details.

2.1.1 OMERO.web

OMERO.web is a web-based client for users who wish to access their data in the browser. This offers a similar view to the OMERO.insight desktop client. Figures OMERO.web user interface and OMERO.web image viewer present the user interface. Developers can use the OMERO.web framework to build customized views.

For more information and guides to using OMERO.web, see our help website\(^3\).

2.1.2 OMERO.insight

OMERO.insight provides a number of tools for accessing and using data in an OMERO server. Figures OMERO.insight and OMERO.insight ImageViewer and Measurement Tool present the user interface. To find out more, see the OMERO.insight user guides\(^4\).

Among many features, the noteworthy OMERO.insight elements are:

- DataManager, a traditional tree-based view of the data hierarchies in an OMERO server. DataManager supports access to all image metadata, annotations, tags

---

2.1. OMERO clients overview
OMERO Documentation, Release 5.2.6

2.1.3 OMERO.importer

The OMERO.importer is part of the OMERO.insight client, but can also run as a stand-alone application. The OMERO.importer allows the import of proprietary image data files from a filesystem accessed from the user’s computer to a running OMERO server. This tool uses a standard file browser to select the image files for import into an OMERO server.

The tool uses Bio-Formats for translation of proprietary file formats in preparation for upload to an OMERO Server. Visit Supported Formats for a detailed list of supported formats.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

2.2 Command Line Interface as an OMERO client

The CLI (Command Line Interface) is a set of Python based system-administration, deployment and advanced user tools. Most of commands work remotely so that the CLI can be used as a client against an OMERO server.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

5http://help.openmicroscopy.org/
6http://www.openmicroscopy.org/site/support/bio-formats/supported-formats.html
7http://www.python.org
2.2.1 Installation

Check you have Python\(^8\) installed by typing:

```bash
$ python --version
Python 2.7.9
```

Additionally, Ice\(^9\) must be installed on your machine:

```bash
$ python -c "import Ice"
```

Check the server requirements for the minimum and maximum supported versions.

The CLI is currently bundled:

- with the OMERO.server including all functionalities of the CLI
- with the OMERO.python including all functionalities of the CLI except for the import functionality

Download the version corresponding to your system from the OMERO downloads\(^10\) page.

**Note:** The CLI is bundled with the OMERO.server but that does not imply you must use that directory as a server. You can download the server zip to a number of machines and use the CLI commands from each machine to access an existing OMERO instance.

Once the correct bundle is downloaded, the Python libraries of the CLI are located under the `lib/python` directory and the executable is under the `bin` directory. To start the CLI in shell mode:

```bash
$ bin/omero
OMERO Python Shell. Version 5.2.6-ice35
Type `help` for more information, `quit` or Ctrl-D to exit
omero>
```

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

---

\(^8\)http://www.python.org

\(^9\)https://zeroc.com

\(^10\)http://downloads.openmicroscopy.org/latest/omero/
2.2.2 Overview

Command line help

The CLI is divided into several commands which may themselves contain subcommands. You can investigate the various commands available using the \(-h\) or \(--help\) option:

$ bin/omero -h

Again, you can use \(-h\) repeatedly to get more details on each of these sub-commands:

$ bin/omero admin -h
$ bin/omero admin start -h

The \texttt{omero help} command can be used to get info on other commands or options.

\texttt{command}

Display the information on a particular command:

$ bin/omero help admin  # same as bin/omero admin -h

In addition to the CLI commands which can be listed using \(--list\), \texttt{omero help} can be used to retrieve information about the \texttt{debug} and \texttt{env} options:

$ bin/omero help debug  # display help about debugging options
$ bin/omero help env    # display help about environment variables

\texttt{--all}

Display the help for all available commands and options

\texttt{--recursive}

Recursively display the help of commands and/or options. This option can be used with either the \texttt{command} or the \texttt{--all} option:

$ bin/omero help --all --recursive
$ bin/omero help user --recursive

\texttt{--list}

Display a list of all available commands and subcommands

Command line workflow

There are three ways to use the command line tools:

1. By explicitly logging in to the server first i.e. by creating a session using the \texttt{omero login} command, e.g.:

   $ bin/omero login username@servername:4064
   Password:

   During login, a session is created locally on disk and will remain active until you logout or it times out. You can then call the desired command, e.g. the \texttt{omero import} command:

   $ bin/omero import image.tiff
2. By passing the session arguments directly to the desired command. Most commands support the same arguments as `omero login`:

```
$ bin/omero -s servername -u username -p 4064 import image.tiff
Password:
```

The `--sudo` option is available to all commands accepting connection arguments. For instance to import data for user `username`:

```
$ bin/omero import --sudo root -s servername -u username image.tiff
Password for owner:
```

3. By calling the desired command without login arguments. You will be asked to login:

```
$ bin/omero import image.tiff
Server: [servername]
Username: [username]
Password:
```

Once you are done with your work, you can terminate the current session if you wish using the `omero logout` command:

```
$ bin/omero logout
```

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 2.2.3 Import images

The CLI import command allows you to import images to an OMERO.server from the command line, and is ideally suited for anyone wanting to use a shell-scripted or web-based front-end interface for importing. Based upon the same set of libraries as the standard importer, the command line version supports the same file formats and functions in much the same way. Visit Supported Formats for a detailed list of supported formats.

**Import command**

To import a file `image.tif`, use:

```
$ bin/omero import image.tif
```

Some of the options available to the import command are:

```
--help
Display the help for the import command

--java-help
Display the help for the Java options of the import command

Java options can be passed after --, e.g.:

    bin/omero import image.tif -- --name=test --description=description

```

```
--advanced-help
Display the advanced help for the import command

```

11 [http://www.openmicroscopy.org/site/support/bio-formats/supported-formats.html](http://www.openmicroscopy.org/site/support/bio-formats/supported-formats.html)
Some advanced import options are described in the In-place import section.

-f
Display all the files that would be imported, then exit:

$ bin/omero import -f image.tif
$ bin/omero import -f images_folder

This will output a list of all the files which would be imported in groups separated by “#” comments. Note that this usage does not require a running server to be available.

--depth DEPTH
Set the number of directories to scan down for files (default: 4):

$ bin/omero import --depth 1 images_folder

--skip SKIP
Specify optional step to skip during import.

The import of very large datasets like High-Content Screening data or SPIM data can be time and resource consuming both at the client and at the server level. This option allows the disabling of some non-critical steps and thus faster import of such datasets. The caveat associated with its usage is that some elements are no longer generated at import time. Some of these elements, like thumbnails, will be generated at runtime during client access. Available options that can be skipped are currently:

all  Skip all optional steps described below
checksum  Skip checksum calculation on image files before and after transfer
          This option effectively sets the --checksum_algorithm to use a fast algorithm, File-Size-64, that considers only file size, not the actual file contents.
minmax  Skip calculation of the minima and maxima pixel values
          This option will also skip the calculation of the pixels checksum. Recalculating minima and maxima pixel values post-import is currently not supported. See Calculation of minima and maxima pixel values for more information.
thumbnails  Skip generation of thumbnails
          Thumbnails will usually be generated when accessing the images post-import via the OMERO clients.
upgrade  Skip upgrade check for Bio-Formats
Example of usage:

$ bin/omero import large_image --skip all
$ bin/omero import large_image --skip minmax

Multiple import steps can be skipped by supplying multiple arguments:

$ bin/omero import large_image --skip checksum --skip minmax

-T, --target TARGET
Select an import target. See Using import targets for more information.

--debug DEBUG
Set the debug level for the command line import output:

$ bin/omero import images_folder --debug WARN

--report
Report emails to the OME team. This flag is mandatory for the --upload and --logs arguments.
--email EMAIL
Set the contact email to use when reporting errors. This argument should be used in conjunction with the --report and --upload or --logs arguments.

--upload
Upload broken files and log file (if any) with report
The following command would import a broken image and upload it together with the import log if available in case of failure:

$ bin/omero import broken_image --report --upload --email my.email@domain.com

--logs
Upload log file (if any) with report
The following command would import a broken image and upload only the import log if available in case of failure:

$ bin/omero import broken_image --report --logs --email my.email@domain.com

Command Line Importer
The CLI import plugin calls the ome.formats.importer.cli.CommandLineImporter Java class. The Linux OMERO.importer also includes an importer-cli shell script allowing calls to the importer directly from Java. Using importer-cli might look like this:

./importer-cli -s localhost -u user -w pass image.tif

To use the ome.formats.importer.cli.CommandLineImporter class from java on the command line you will also need to include a classpath to the required support jars. Please inspect the importer-cli script for an example of how to do this.

The Command Line Importer tool takes a number of mandatory and optional arguments to run. These options will also be displayed on the command line by passing no arguments to the importer:

Usage: importer-cli [OPTION]... [path [path ...]]...
       or: importer-cli [OPTION]... -

Import any number of files into an OMERO instance.
If "-" is the only path, a list of files or directories is read from standard in. Directories will be searched for all valid imports.

Session arguments:
Mandatory arguments for creating a session are 1- either the OMERO server hostname, username and password or 2- the OMERO server hostname and a valid session key.
- SERVER OMERO server hostname
- u USER OMERO username
- w PASSWORD OMERO password
- k KEY OMERO session key (UUID of an active session)
- p PORT OMERO server port (default: 4064)

Naming arguments:
All naming arguments are optional
- n NAME Image or plate name to use
- x DESCRIPTION Image or plate description to use
- name NAME Image or plate name to use
- description DESCRIPTION Image or plate description to use

Optional arguments:
OMERO Documentation, Release 5.2.6

Examples:

$ importer-cli -s localhost -u user -w password -d 50 foo.tiff
$ importer-cli -s localhost -u user -w password -d Dataset:50 foo.tiff
$ importer-cli -f foo.tiff
$ importer-cli -s localhost -u username -w password -d 50 --debug ALL foo.tiff

For additional information, see:
http://www.openmicroscopy.org/site/support/omero5.2/users/cli/import.html
Report bugs to <ome-users@lists.openmicroscopy.org.uk>

See also:
Advanced import scenarios

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

2.2.4 Using import targets

The CLI import options –d or –r can be used to specify, respectively, the import target Dataset or Screen by ID. The option –T, --target adds more ways of specifying the import target.

The general form of the target argument is:

<action or Class>[:<discriminator>]:<pattern>

where the discriminator is optional. Thus a target must contain one or two colons. Any further colons will be read as part of the pattern. If the discriminator is omitted a default will be used depending on the action or Class. Currently the following actions and classes are supported: Dataset, Screen and regex.

Importing to a Dataset or Screen

For Dataset and Screen the currently supported discriminators are name and id. If the discriminator is omitted the default used is id. So:

$ bin/omerom import ~/images/dv/SMN10ul03_R3D_D3D.dv -T Dataset:id:2
$ bin/omerom import ~/images/dv/SMN10ul03_R3D_D3D.dv -T Dataset:2
$ bin/omerom import ~/images/dv/SMN10ul03_R3D_D3D.dv -d 2

will all have the same effect of importing the image to the Dataset with ID 2.

The name discriminator can be used to select the target by name, and so:
OMERO Documentation, Release 5.2.6

$ bin/omero import ~/images/dv/SMN10ul03_R3D_D3D.dv -T Dataset:name:Sample01

will import the image to the Dataset with name Sample01. If more than one Dataset exists with the specified name the most recently created will be used. If no Dataset exists with the specified name a new Dataset will be created for the import.

The choice of Dataset can be specified by adding a qualifying character to the discriminator: + to use the most recent, – to use the oldest, % to only import if there is a unique target or @ to create a new container even if one with the correct name already exists.

For example:

$ bin/omero import ~/images/dv/SMN10ul03_R3D_D3D.dv -T Dataset:+name:Samples
$ bin/omero import ~/images/dv/SMN10ul03_R3D_D3D.dv -T Dataset:-name:Samples
$ bin/omero import ~/images/dv/SMN10ul03_R3D_D3D.dv -T Dataset:%name:Samples
$ bin/omero import ~/images/dv/SMN10ul03_R3D_D3D.dv -T Dataset:@name:Samples

The first case is equivalent to the previous example, the most recent Dataset will be used. In the second case the oldest Dataset will be used. In the third case the import should fail if multiple datasets with that name exist. In the first three cases a new Dataset will be created if none exists. In the last case a new Dataset should be created even if one or more already exist.

If the name contains spaces or other characters that cannot be used on the command line the pattern should be enclosed in quotes:

$ bin/omero import ~/images/dv/SMN10ul03_R3D_D3D.dv -T Dataset:name:"New Dataset"

To import a plate to a Screen target the same syntax can be used as in all the examples above, for example:

$ bin/omero import ~/images/bd-pathway/2015-12-01_000/ -T Screen:+name:Pathway

Importing using regular expressions

The local path of the file to be imported can be used to specify the target Dataset or Screen using a regular expression using the action regex. For this action the only discriminator is name and if the discriminator is omitted the qualified form of this +name will be used. The sequence (?<Container1>.*?) is a named-capturing group used to specify the Dataset or Screen name in the regular expression, the specific name Container1 must be used here. For example:

$ bin/omero import ~/images/dv/SMN10ul03_R3D_D3D.dv -T "regex:^.*images/(?<Container1>.*?)"

would use a Dataset with name being the path following images/, in this case dv.

The name discriminator can be explicitly used and, as in the previous section, also qualified:

$ bin/omero import ~/images/dv/SMN10ul03_R3D_D3D.dv -T "regex:+name:^.*images/(?<Container1>.*?)"
$ bin/omero import ~/images/dv/SMN10ul03_R3D_D3D.dv -T "regex:-name:^.*images/(?<Container1>.*?)"
$ bin/omero import ~/images/dv/SMN10ul03_R3D_D3D.dv -T "regex:%name:^.*images/(?<Container1>.*?)"
$ bin/omero import ~/images/dv/SMN10ul03_R3D_D3D.dv -T "regex:@name:^.*images/(?<Container1>.*?)"

These each work in the same way as the previous Dataset examples.

In some cases the importable files may be in nested directories, this is often the case with plates and some multi-image formats. A regular expression can be used to pick a higher level directory as the Screen or Dataset name. For example, if several BD Pathway HCS files are under the following paths:

~/images/bd-pathway/week-1/2015-12-01_000/
~/images/bd-pathway/week-2/2015-12-09_000/
~/images/bd-pathway/week-2/2015-12-11_000/

2.2. Command Line Interface as an OMERO client
and the intended Screens for the import are week-1 and week-2 then the following could be used:

```
$ bin/omero import ~/images/bd-pathway/ -T "regex:+name:^.*bd-pathway/\<Container1>[^/]*\>/.*
```

which would import one Plate into the Screen week-1 and two Plates into the Screen week-2, creating those Screens if necessary.

A useful way of determining the nested structure to help in constructing regular expressions is the option `-f` which displays the used files but does not import them:

```
$ bin/omero import -f ~/images/bd-pathway/week-1
```

which shows that all the files for one particular Plate from the example above are under:

```
```

For more information on the regular expression syntax that can be used in templates see: java.util.regex.Pattern documentation\(^\text{12}\).

\textbf{Note: This documentation is for OMERO 5.2.} This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

\subsection*{2.2.5 Manage sessions}

The \texttt{omero sessions} plugin manage user sessions stored locally on disk. Several sessions can be active simultaneously, but only one will be used for a single invocation of \texttt{bin/omero}:

```
$ bin/omero sessions -h
```

\subsubsection*{Login}

The \texttt{omero login} command is a shortcut for the \texttt{omero sessions login} subcommand which creates a connection to the server. If no argument is specified, the interface will ask for the connection credentials:

```
$ bin/omero login
```

Some of the options available to the \texttt{omero login} command are:

\begin{itemize}
  \item \texttt{connection} Pass a connection string under the form \texttt{[USER@]SERVER[:PORT]} to instantiate a connection:
\end{itemize}

\(^{12}\text{http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html}\)
$ bin/omero login username@servername
Password:
$ bin/omero login username@servername:14064
Password:

-s SERVER, --server SERVER
Set the name of the server to connect to:

$ bin/omero login -s servername
Username: [username]

-u USER, --user USER
Set the name of the user to connect as:

$ bin/omero login -u username -s servername
Password:

-p PORT, --port PORT
Set the port to use for connection. Default: 4064:

$ bin/omero login -u username -s servername -p 14064
Password:

-g GROUP, --group GROUP
Set the group to use for initializing a connection:

$ bin/omero login -u username -s servername -g my_group
Password:

-k KEY, --key KEY
Use a valid session key to join an existing connection.
This option only requires a server argument:

$ bin/omero login servername -k 22fccb8b-d04c-49ec-9d52-116a163728ca

-w PASSWORD, --password PASSWORD
Set the password to use for the connection.

--sudo ADMINUSER
Create a connection as another user.
The sudo functionality is available to administrators as well as group owners:

$ bin/omero login --sudo root -s servername -u username
Password for root:
$ bin/omero login --sudo owner -s servername -u username
Password for owner:

Multiple sessions

Stored sessions can be listed using the omero sessions list command:
Session keys can then be reused to switch between stored sessions using the `omero login -k` option:

```bash
$ bin/omero sessions login -k 22fccb8b-d04c-49ec-9d52-116a163728ca
Server: [localhost]
Joined session 1f800a16-1dc2-407a-8a85-fb44005306be (root@localhost:4064).
```

Sessions directory

By default sessions are saved locally on disk under the OMERO user directory located at `~/.omero/sessions`. The location of the current session file can be retrieved using the `omero sessions file` command:

```bash
$ bin/omero sessions file
/Users/ome/omero/sessions/localhost/root/aec828e1-79bf-41f3-91e6-a4ac76ff1cd5
```

To customize the OMERO user directory, use the `OMERO_USERDIR` environment variable:

```bash
$ export OMERO_USERDIR=/tmp/omero_dir
$ bin/omero login root@localhost:4064 -w omero
Created session bf7b9fee-5e3f-40fa-94a6-1e23ceb43dbd (root@localhost:4064). Idle timeout: 10.0 min.
$ bin/omero sessions file
/tmp/omero_dir/omero/sessions/localhost/root/bf7b9fee-5e3f-40fa-94a6-1e23ceb43dbd
$ bin/omero logout
```

If you want to use a custom directory for sessions exclusively, use the `OMERO_SESSIONDIR` environment variable:

```bash
$ export OMERO_SESSIONDIR=/tmp/my_sessions
$ bin/omero login root@localhost:4064 -w omero
Created session bf7b9fee-5e3f-40fa-94a6-1e23ceb43dbd (root@localhost:4064). Idle timeout: 10.0 min.
$ bin/omero sessions file
/tmp/my_sessions/localhost/root/bf7b9fee-5e3f-40fa-94a6-1e23ceb43dbd
$ bin/omero logout
```

Note: The `OMERO_SESSION_DIR` environment variable introduced in 5.1.0 to specify a custom sessions directory is deprecated in 5.1.1 and above in favor of `OMERO_SESSIONDIR`.

If you have been using `OMERO_SESSION_DIR` and want to upgrade your custom sessions directory without losing locally stored sessions:

- either set `OMERO_SESSIONDIR` to point at the same location as `OMERO_SESSION_DIR/omero/sessions`
- or move all local sessions stored under the `OMERO_SESSION_DIR/omero/sessions` directory under the `OMERO_SESSION_DIR` directory and replace `OMERO_SESSION_DIR` by `OMERO_SESSIONDIR`.

---

2.2. Command Line Interface as an OMERO client

Switching current group

The `sessions group` command can be used to switch the group of your current session:

```
$ bin/omero group list  # list your groups
$ bin/omero sessions group 2  # switch to group by ID or Name
```

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

2.2.6 Manage tags

The `omero tag` subcommands manage the creation, linking and listing of tag annotations. All subcommands can be listed using the `-h` option:

```
$ bin/omero tag -h
```

Create tags

To create a new tag annotation, use the `omero tag create` command:

```
$ bin/omero tag create
Please enter a name for this tag: my_tag
```

To create a tag set containing two existing tags of known identifiers 1259 and 1260, use the `omero tag createset` command:

```
$ bin/omero tag createset --tag 1259 1260
Please enter a name for this tag set: my_tag_set
```

For both tags and tag sets, the name and an optional description can be passed using the `--name` and `--desc` options:

```
$ bin/omero tag create --name my_tag --desc 'description of my_tag'
$ bin/omero tag createset --tag 1259 1260 --name my_tag_set --desc 'description of my_tag_set'
```

List tags

To list all the tags owned by the current user, use the `omero tag list` command:

```
$ bin/omero tag list
+- 1261:'my_tag_set'
|\   
| + 1259:'my_tag'
| + 1260:'my_tag_2'
+- 1264:'my_tag_set_2'
|\   
| + 1260:'my_tag_2'
| + 1263:'my_tag_4'

Orphaned tags:
> 1262:'my_tag_3'
```
To list all the tag sets owned by the current user, use the `omero tag listsets` command:

```
$ bin/omero tag listsets
--------|---------------------------------------------------------------
ID    |Name
-------|---------------------------------------------------------------
1261   |my_tag_set
1264   |my_tag_set_2
--------|---------------------------------------------------------------
```

### Link tags

Tags can be linked to objects on the server using the `omero tag link` command. The object must be specified as `object_type:object_id`. To link the tag of identifier 1260 to the Image of identifier 1000, use:

```
$ bin/omero tag link Image:1000 1260
```

### Delete tags

Tags can be deleted using the `omero delete` command. The tag or tag set must be specified as `TagAnnotation:tag_id`. To delete tag 123 use:

```
$ bin/omero delete TagAnnotation:123
```

By default the tags within a tag set will not be deleted with the tag set. To delete any included tags use the `--include` option:

```
$ bin/omero delete TagAnnotation:123 --include TagAnnotation
```

### See also:

*Deleting objects*

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 2.2.7 Moving objects between groups

The `omero chgrp` command moves objects between groups. Further help is available using the `-h` option:

```
$ bin/omero chgrp -h
```

The `chgrp` command will remove entire graphs of objects based on the IDs of the topmost objects. The command can be modified to include the movement of objects that would, by default, be excluded or exclude objects that would, by default, be included using the `--include` and `--exclude` options.

It is also possible to move objects lower in the hierarchy by specifying the type and ID of a topmost object and the type of the lower object. For instance, moving all of the images under a given project.

By default the command confirms the movement of the target objects but it can also provide a detailed report of all the moved objects via a `--report` option. A `--dry-run` option can be used to report on what objects would be moved without actually moving them.
Examples

Basic move

$ bin/omero chgrp 5 OriginalFile:101
$ bin/omero chgrp Group:5 Project:51
$ bin/omero chgrp ExperimenterGroup:5 Project:51
$ bin/omero chgrp lab_group Project:51

In the first line, the original file with ID 101 will be moved to the group with ID 5. In the second and third, project 51 will be moved to group 5 including any datasets inside only that project and any images that are contained within moved datasets only. If group 5 is named ‘lab_group’ then the last line will have the same effect as the previous two. Note that any linked annotations will also be moved.

Moving multiple objects

Multiple objects can be specified with each type being followed by an ID or a comma-separated list of IDs. The order of objects or IDs is not significant, thus all three calls below are identical in moving project 51 and datasets 53 and 54 to group 5.

$ bin/omero chgrp 5 Project:51 Dataset:53,54
$ bin/omero chgrp 5 Dataset:54,53 Project:51
$ bin/omero chgrp 5 Dataset:53 Project:51 Dataset:54

Moving lower level objects

To move objects below a specified top-level object the following form of the object specifier is used.

$ bin/omero chgrp 5 Project/Dataset/Image:51

Here the all of images under the project 51 would be moved. It is not necessary to specify intermediate objects in the hierarchy and so:

$ bin/omero chgrp 5 Project/Image:51

would have the same effect as the call above.

Including and excluding objects

--include
Linked objects that would not ordinarily be moved can be included in the move using the –include option:

$ bin/omero chgrp 5 Image:51 --include Annotation

This call would move any annotation objects linked to the image.

--exclude
Linked objects that would ordinarily be moved can be excluded from the move using the –exclude option:

$ bin/omero chgrp 5 Project:51 --exclude Dataset

This will move project 51 but not any datasets contained in that project.
The two options can be used together:

$ bin/omero chgrp 5 Project/Dataset:53 --exclude Image --include FileAnnotation

This will move any datasets under project 53, that are not otherwise contained elsewhere, excluding any images in those datasets but including any file annotations linked to the moved datasets. In this case the images that are not otherwise contained in datasets will be orphaned.

Further options

--ordered
Move the objects in the order specified.

Normally all of the specified objects are grouped into a single move command. However, each object can be moved separately and in the order given. Thus:

$ bin/omero chgrp 5 Dataset:53 Project:51 Dataset:54 --ordered

would be equivalent to making three separate calls:

$ bin/omero chgrp 5 Dataset:53
$ bin/omero chgrp 5 Project:51
$ bin/omero chgrp 5 Dataset:54

--report
Provide a detailed report of what is moved:

$ bin/omero chgrp 5 Project:502 --report

--dry-run
Run the command and report success or failure but does not move the objects. This can be combined with the --report to provide a detailed confirmation of what would be moved before running the move itself.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

2.2.8 Deleting objects

The omero delete command deletes objects. Further help is available using the -h option:

$ bin/omero delete -h

The delete command will remove entire graphs of objects based on the IDs of the topmost objects. The command can be modified to include the deletion of objects that would, by default, be excluded or exclude objects that would, by default, be included using the --include and --exclude options.

Additionally, objects of the three annotation types, FileAnnotation, TagAnnotation and TermAnnotation are not deleted by default when the objects to which they are linked are deleted.

It is also possible to delete objects lower in the hierarchy by specifying the type and ID of a topmost object and the type of the lower object. For instance, deleting all of the images under a given project.

By default the command confirms the deletion of the target objects but it can also provide a detailed report of all the deleted objects via a --report option. A --dry-run option can be used to report on what objects would be deleted without actually deleting them.
Examples

Basic delete

$ bin/omero delete OriginalFile:101
$ bin/omero delete Project:51

In the first line, the original file with ID 101 will be deleted. In the second, the project with ID 51 will be deleted including any datasets inside only that project and any images that are contained within moved datasets only. Note that any linked file, tag or term annotations will not be deleted.

Deleting multiple objects

Multiple objects can be specified with each type being followed by an ID or a comma-separated list of IDs. The order of objects or IDs is not significant, thus all three calls below are identical in deleting project 51 and datasets 53 and 54.

$ bin/omero delete Project:51 Dataset:53,54
$ bin/omero delete Dataset:54,53 Project:51
$ bin/omero delete Dataset:53 Project:51 Dataset:54

Deleting lower level objects

To delete objects below a specified top-level object the following form of the object specifier is used.

$ bin/omero delete Project/Dataset/Image:51

Here the all of images under the project 51 would be deleted. It is not necessary to specify intermediate objects in the hierarchy and so:

$ bin/omero delete Project/Image:51

would have the same effect as the call above. Links can also be deleted and so:

$ bin/omero delete Project/DatasetImageLink:51 Dataset/DatasetImageLink:53

would effectively orphan all images under project 51 and dataset 53 that are not also under other datasets.

Including and excluding objects

--include

Linked objects that would not ordinarily be deleted can be included in the delete using the --include option:

$ bin/omero delete Image:51 --include FileAnnotation,TagAnnotation,TermAnnotation

As mentioned above these three annotation types are not deleted by default and so this call overrides that default by including any of the three annotation types in the delete:

$ bin/omero delete Image:51 --include Annotation

This call would also delete any annotation objects linked to the image.
--exclude
Linked objects that would ordinarily be deleted can be excluded from the delete using the --exclude option:

$ bin/omero delete Project:51 --exclude Dataset

This will delete project 51 but not any datasets contained in that project.

The two options can be used together:

$ bin/omero delete Project/Dataset:53 --exclude Image --include FileAnnotation

This will delete any datasets under project 53, that are not otherwise contained elsewhere, excluding any images in those datasets but including any file annotations linked to the deleted datasets. In this case the images that are not otherwise contained in datasets will be orphaned.

For an example on deleting tags directly see Delete tags.

Further options

--ordered
Delete the objects in the order specified.

Normally all of the specified objects are grouped into a single delete command. However, each object can be deleted separately and in the order given. Thus:

$ bin/omero delete Dataset:53 Project:51 Dataset:54 --ordered

would be equivalent to making three separate calls:

$ bin/omero delete Dataset:53
$ bin/omero delete Project:51
$ bin/omero delete Dataset:54

--report
Provide a detailed report of what is deleted:

$ bin/omero delete Project:502 --report
... omero.cmd.Delete2 Project 502... ok
Steps: 3
Elapsed time: 0.597 secs.
Flags: []
Deleted objects
Dataset:603
DatasetImageLink:303
Project:503
ProjectDatasetLink:353
Channel:203
Image:503
LogicalChannel:203
OriginalFile:460,459
Pixels:253
Fileset:203
FilesetEntry:253
FilesetJobLink:264,265,262,263,261
IndexingJob:315
JobOriginalFileLink:303
MetadataImportJob:312

2.2. Command Line Interface as an OMERO client
--dry-run

Run the command and report success or failure but do not delete the objects. This can be combined with the --report to provide a detailed confirmation of what would be deleted before running the delete itself.

See also:

Command Line Interface as an OMERO admin tool  System administrator documentation for the Command Line Interface
Command Line Interface as an OMERO development tool  Developer documentation for the Command Line Interface
ADDITONAL RESOURCES

• About OMERO\(^1\) introduces OMERO for new users, while the Features List\(^2\) provides an overview of the platform features with those that are new for OMERO 5.2 highlighted.

• Workflow-based user assistance guides are provided on our [help website]\(^3\).

• As OMERO is an open source project with developers and users in many countries, [connecting to the community](http://help.openmicroscopy.org/) can provide you with a wealth of experience to draw on for help and advice.

• Our partners within the OME consortium are working on integrating additional functions and modules with OMERO. See the [Partner Projects](http://www.openmicroscopy.org/site/products/partner/) page for details of the latest extensions which could help OMERO meet your research needs more fully.

• You can also extend the functionality of OMERO using OMERO.scripts, our version of plugins. Guides to some of the scripts which ship with OMERO releases are already provided, but you can also check out our [Script Sharing](http://www.openmicroscopy.org/site/community/scripts) page to find extra ones.

---

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 3.1 Community support

The [Open Microscopy Environment](http://www.openmicroscopy.org/site) provides a number of resources for both our user and developer communities to assist in use and development of our software. Contributions through our mailing lists and forums are always welcome.

#### 3.1.1 Web


#### 3.1.2 Mailing lists

The following lists are provided:

• **ome-users**\(^7\) – support with installation and general use or miscellaneous queries, as well as bug reporting

• **ome-devel**\(^8\) – development discussion and support

**Note:** You can email these lists without subscribing but posts by non-members will be moderated before going out to all members. We endeavor to approve all genuine posts within 24 hours on working days, but we encourage you to subscribe to take advantage of our active and supportive community.

---

\(^1\)[http://www.openmicroscopy.org/site/products/omero/]
\(^2\)[http://help.openmicroscopy.org/]
\(^3\)[http://lists.openmicroscopy.org.uk/mailman/listinfo/ome-users/]
\(^4\)[http://www.openmicroscopy.org/site/products/partner/]
\(^5\)[http://www.openmicroscopy.org/site/community/scripts]
\(^6\)[http://www.openmicroscopy.org/site]
\(^7\)[http://lists.openmicroscopy.org.uk/mailman/listinfo/ome-users/]
\(^8\)[http://lists.openmicroscopy.org.uk/mailman/listinfo/ome-devel/]
3.1.3 Forums

Discussion on a number of topics is also available through our forums. Forums include:

- OME Announcements
- OMERO
  - User Discussion
  - Installation and Deployment
  - Developer Discussion
- Bio-Formats
  - User Discussion
- OME Data Model
  - User Discussion and Suggestions

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

3.2 What’s new for OMERO 5.2 for users

Updates and new features for OMERO 5.2 include:

- improved support for screen data formats (amongst others) and import times for images with many ROIs
- reworked display of metadata and annotations in the right hand pane of both clients
- improved support for attaching analytical results
- significant improvement in client performance including for loading data and drag and drop actions
- the size limit for exporting images is now 12k x 12k pixels

Note that you can no longer launch the desktop clients via Java Web Start as we have dropped support for this due to technical issues beyond our control.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

3.3 OMERO version history

3.3.1 5.2.6 (October 2016)

This is a bug-fix release focusing on services closure and a DB upgrade fix. Improvements include:

9 http://www.openmicroscopy.org/community/
12 http://www.openmicroscopy.org/community/viewforum.php?f=4
• fixed closure of session in Java when using Ice 3.5
• fixed memory leak where services were not correctly closed
• added a DB upgrade patch to fix errors only affecting DBs that have been upgraded from OMERO 4.4
• fixed a MATLAB regression introduced in 5.2.2, casting error.
• fixed error in logs on getProjectedThumbnail

Support for OMERO.web deployment using Apache has also been deprecated and is likely to be removed during the 5.3.x line.

### 3.3.2 5.2.5 (August 2016)

This is a security release to fix the access privileges of the share function, which were potentially allowing users to access private data belonging to other users via the API.

See [2016-SV2-share](http://www.openmicroscopy.org/info/vulnerabilities/2016-SV2-share) for details. Shares will now respect user privileges as set by the group permission level. Note that Shares now only support images even when used via the API.

It is highly recommended that you upgrade your server. For those not in a position to do so as a matter of urgency, a workaround is provided which deletes all shares and disables their creation.

### 3.3.3 5.2.4 (May 2016)

This is a security release to fix the cleanse.py script used by the “bin/omero admin cleanse” command, which was not properly respecting user permissions and may lead to data loss.

See [2016-SV1-cleanse](http://www.openmicroscopy.org/info/vulnerabilities/2016-SV1-cleanse) for details. The script and command have now been made admin-only.

It is highly suggested that you upgrade your server or apply the patch available from the security page.

### 3.3.4 5.2.3 (May 2016)

A bug-fix release. Improvements include:

• fixed problem with float images
• all scripts currently exposed to users via our website have been reviewed and fixed where necessary so they are all now 5.2.x compatible, and a new omero-install workflow has been developed to ensure these are reviewed regularly going forward
• better support for metadata annotations in clients including tag/tagset support and performance issues
• fixes in OMERO.web for deleting MIFs
• improvements to the navigation of large datasets and display of plates in OMERO.web
• other OMERO.web bug fixes
• OMERO.insight and CLI import improvements
• other OMERO.insight bug fixes, including for downloading data

Developer updates include:

• Java gateway improvements

System administrator updates include:

• Ice 3.6.2 support for UNIX-like systems, including specific installation walkthroughs
• redis support for websessions caching
• a fix to allow OMERO.web to be run in a Docker container
• improved OMERO.web configuration

---

20http://www.openmicroscopy.org/info/vulnerabilities/2016-SV2-share
21http://www.openmicroscopy.org/info/vulnerabilities/2016-SV1-cleanse
• warnings added regarding the end of Windows support in the 5.3.0 release\textsuperscript{22} (note that we will be preparing a guide for migrating from Windows for existing servers and adding it to the documentation as soon as we can)

This release also upgrades the version of Bio-Formats which OMERO uses to 5.1.9. See what's-new for more information\textsuperscript{23}.

3.3.5 5.2.2 (February 2016)

A bug-fix release which also introduces some new client features. Improvements include:

• display of ROI masks in OMERO.web image viewer
• display of OMERO.tables data for Wells in the OMERO.web right hand panel
• ‘Populate Metadata’ script to enable generation of OMERO.tables for Wells is now usable from both OMERO.web and OMERO.insight (note this is still in development and has some limitations)
• measurement tool fixes
• fixed pixel size metadata and scalebar in OMERO.web image viewer for images with pixel size units other than micrometer
• fixed OMERO.web handling of turning off interpolation of pixels
• previous and next buttons fixed in OMERO.web image viewer
• delete and change group performance improvements
• better handling of dates in search
• client support for map annotations in OME-TIFF
• disabled orphaned container feature
• OMERO.web clean-up to remove obsolete volume viewer

Developer updates include:

• Python API examples for creating Polygon and Mask shapes
• Python API example for “Populate Metadata” to create OMERO.tables for Wells
• OMERO.tables documentation extended
• updated ‘What’s New for developers’ to clarify that pojos has been renamed as omero.gateway.model
• dynamic scripts functionality documented
• dynamic loading of omero.client server settings into HTTP sessions

System administrator updates include:

• clarification of OMERO.web documentation for nginx deployment, including an experimental solution to resolve download issues
• documentation of hard-linking issues for in-place import on linux systems

Note that the OMERO Virtual Appliance has been discontinued and will not be updated for version 5.2.2 or any later releases.

This release also upgrades the version of Bio-Formats which OMERO uses to 5.1.8. See what's-new for more information\textsuperscript{24}.

3.3.6 5.2.1 (December 2015)

A bug-fix release focusing on improving installation documentation and workflows. Other improvements include:

• bug fix for missing hierarchy when viewing High Content Screening data
• improvements to the right-hand panel in OMERO.insight
• measurement tool fixes
• OMERO.web fix for displaying size units

\textsuperscript{22}http://blog.openmicroscopy.org/tech-issues/future-plans/deployment/2016/03/22/windows-support/
\textsuperscript{23}https://www.openmicroscopy.org/site/support/bio-formats5.1/about/whats-new.html
\textsuperscript{24}https://www.openmicroscopy.org/site/support/bio-formats5.1/about/whats-new.html
System administrator updates include:

- improved installation documentation, including detailed walkthroughs for specific OS
- OMERO.web deployment fixes

Developer updates include:

- OMERO Javadocs now link to the relevant version of Bio-Formats Javadocs for inherited methods
- clean-up of server dependencies
- jstree clean-up
- CLI graph operation improvements for deleting
- minimal-omero-client and pom-omero-client clean-up

This release also upgrades the version of Bio-Formats which OMERO uses to 5.1.7.

### 3.3.7 5.2.0 (November 2015)

A full, production-ready release of OMERO 5.2.0; dropping support for Java 1.6; featuring major upgrading of OMERO.web; re-working of the Java Gateway; and introducing new user features and many fixes and performance improvements:

- improved support for many file formats via Bio-Formats 5.1.5
- faster import for images with a large number of ROIs
- performance improvements for OMERO.web including faster data tree loading
- Java Web Start has been dropped, it is no longer possible to launch the desktop clients from the web
- reworked display of metadata and annotations in both UI clients
- many bugs fixed

Developer and system administrator updates include:

- the OMERO web framework no longer bundles a copy of the Django package, this dependency must be installed manually
- updated jstree to 3.08 and now using json for all tree loading to substantially improve performance
- removed FastCGI support, OMERO.web can be deployed using WSGI
- configuration property `omero.graphs.wrap` which allowed switching back to the old server code for moving and deleting data has now been removed. You should migrate to using the new graph request operations before 5.3 when the old request operations will be removed
- introduced new Java Gateway to facilitate the development of Java applications
- aligned OMERO Rect with OME-XML schema for ROI. Clients using the OMERO.blitz server API to work with ROIs will need to be updated

### 3.3.8 5.1.4 (September 2015)

A bug-fix release covering all components. Improvements include:

- channel buttons fixed in OMERO.web
- improved UI experience when moving annotated data between groups in OMERO.web
- improved performance for loading annotations in the right-hand panel of OMERO.web
- much better handling of ROIs covering large planes in OMERO.insight
- rendering setting fixes for copy and paste actions in OMERO.insight
- rendering fixes for floating point data
- Admins can now configure whether the clients interpolate images by default
- better formatting of Delta-T and exposure times in the clients
• directories are now preserved when downloading multiple original files
• various improvements to the OMERO-ImageJ handling of ROIs and measurements, including the ability to name measurement tables
• current session key can now be returned via the CLI
• other CLI improvements including usability of ‘chmod’ for downgrading group permissions, and listing empty tagsets
• added support for groups in OMERO.matlab methods

Developer updates include:
• improvements to web logging to log full request and status code
• fixed joda-time version mismatch
• cleanup of old insight code to remove remaining references to OMERO.editor

Support for deployment of OMERO.web using FastCGI has also been deprecated in this release and is scheduled to be removed in 5.2.0. Sysadmins should move to using WSGI instead. We are also intending to stop distributing Java Webstart for launching OMERO.insight from your browser, as security concerns mean browsers are increasingly moving away from supporting this type of application. You can read further information regarding this decision on our Web Start blog post.

3.3.9 5.1.3 (July 2015)

A bug-fix release which also introduces some new functionality. Improvements include:
• tagging actions extended; you can now use tag sets to tag images on import
• tagging ome-tiff images at import has also been fixed
• greatly improved workflow and bug fixes for the Share functionality in OMERO.web which enables you to share images with users outside of your group (including removal of part of the UI)
• group admins and owners can now change ownership of data via the CLI
• better reporting for the ‘delete’ and ‘chgrp’ functionality in the CLI
• fixed display of images in plates with multiple acquisitions
• fixed export of results as .xls files from OMERO.insight
• improved workflow for ImageJ and OMERO interactions
• support for WSGI OMERO.web deployment
• fixed OMERO.mail service for web errors
• fixes for ROI display in OMERO.web (thanks to Luca Lianas of CRS4)
• fixes and workflow improvements for running scripts and script dialogs

Developer updates include:
• OMERO.web clean-up (removal of ‘-locked’)
• reorganization of the server bundle to move various licenses and dependencies under a new ‘share’ folder
• refactoring of ‘Chown2’, ‘Chmod2’, ‘Chgrp2’ and ‘Delete2’
• addition of dynamic scripts
• the ‘rstring’ implementation is now more lenient and should better handle unicode
• Bio-Formats submodule removed from OMERO; decoupling effort means OMERO now consumes the Bio-Formats release build from the artifactory

This release also includes the fix for the Java security issue, as discussed in the recent blog post. Testing suggests this fix should not have any performance implications. You should upgrade your Java version to take advantage of the security fix.
3.3.10 5.1.2 (May 2015)

A bug-fix release which also introduces some new functionality. Improvements include:

- support for Read-Write groups
- the LDAP plugin can now set users as group owners whether on creation or via the improved sync_on_login option
- users logged into the web client can now automatically log in via webstart
- results tables from ImageJ/Fiji can be attached to images in OMERO and the ImageJ/Fiji workflow has been improved
- better delete functionality and warnings in the UI
- improved graph operations like ‘delete’ and ‘chgrp’, as well as the new ‘chmod’ operation (for changing group permissions), are now used across the clients including the CLI
- an API for setting and querying session timeouts is now available via the CLI
- magnification now reflects microscopy values (e.g. 40x) rather than a percentage in both clients
- more readable truncation of file names in the OMERO.insight data tree
- OMERO.web fixes and improvements including:
  - interpolation
  - optimization of plate grid and right-hand panel
  - option to download single original files
  - significant speed-up in loading large datasets
- deployment fixes include:
  - new default permissions on the var/ directory
  - better checks of the Dropbox directory permissions
  - new and some deprecated environment variables
  - a startup check for lock files on NFS
  - use /var/run for omero.fcgi

Critical bugs which were fixed include:

- the in-place import file handle leak (which was a regression in 5.1.1)
- various unicode and unit failures were corrected

3.3.11 5.1.1 (April 2015)

A bug-fix release focusing on user-facing issues and cleaning resources for developers. Improvements include:

For OMERO.web:

- significant review of the web share functionality
- correction of thumbnail refreshing
- fixes to the user administration panel
- fix for embedding of the Javascript image viewer

For OMERO.insight:

- improved open actions
- tidying of the menu structure
- correction of the mouse zoom behavior
- fix for the Drag-n-Drop functionality

Other updates include:
• overhaul of the CLI session log-in logic
• cleaning and testing of all code examples
• further removal of the use of deprecated methods

3.3.12 5.1.0 (April 2015)

A full, production-ready release of OMERO 5.1.0; updating the Data Model to the January 2015 schema, including support for units and new more flexible user-added metadata; and introducing new user features, new supported formats and many fixes and performance improvements:

• support for units throughout the Data Model allowing for example, pixel sizes for electron microscopy to be stored in nanometers rather than being set as micrometers
• new, searchable key-value pairs annotations for adding experimental metadata (replacing OMERO.editor, which has been removed)
• improved workflow for rendering settings in the UI and parity between the clients
• import images to OMERO from ImageJ and save ROIs and overlays from ImageJ to OMERO
• importing as another user, previously only available for administrators, is now usable by group owners as well, allowing you to import data that will then be owned by the user you import it for
• improved performance for moving and deleting data
• removed the auto-levels calculation for initial rendering settings to substantially speed up performance, by using the min/max pixel intensities, or defaulting to full pixel range where min/max is unavailable
• import times are much improved for large datasets such as HCS and SPIM data
• improved performance for many file formats and new supported formats via Bio-Formats (now over 140)
• new OMERO.mail feature lets admins configure the server to email users
• support for configuring the server download policy to control access to original file download for public-facing OMERO.web deployments
• many developer updates such as removal of deprecated methods, and updates to OMERO.web and the C++ implementation (see the 5.1.0-m1 to 5.1.0-m5 developer preview release details below and the ‘What’s New’ for developers page)

3.3.13 5.1.0-m5 (March 2015)

Developer preview release - only intended as a developer preview for updating code before the full public release of 5.1.0. Use at your own risk.

Changes include:

• implementation of OMERO.mail for emailing users via the server
• performance improvements for importing large datasets
• support for limiting the download of original files
• various fixes for searching and filtering map annotations and converting between units
• deprecation of IUpdate.deleteObject API method
• versioning of all JavaScript files to fix browser refresh problems
• clarifying usage of OMERO.web views and templates including RequestContext

3.3.14 5.1.0-m4 (February 2015)

Developer preview release - only intended as a developer preview for updating code before the full public release of 5.1.0. Use at your own risk.

Changes include:
• final Database changes - image.series is now exposed in Hibernate
• improved deletion performance
• client bundle clean-up
• other clean-up work including pep8 and removal of deprecated methods and components
• new Map annotations are now included in the UI and search functionality
• ImageJ plugin updates which allow
  – importing of images and saving ROIs to OMERO from within the plugin
  – viewing images stored in OMERO and their ROIs generated within OMERO from within the plugin
  – updating ROIs on OMERO-stored images within the plugin and saving these back to OMERO without needing to re-import the image
• OMERO.matlab updates re: annotations
• OMERO.tables internal HDF5 format has changed

With thanks to Paul Van Schayck and Luca Lianas for their contributions.

3.3.15 5.0.8 (February 2015)

This is a bug-fix release for one specific issue causing OMERO.insight to crash when trying to open the Projection tab for an image with multiple z-stacks.

3.3.16 5.0.7 (February 2015)

This is a bug-fix release covering a number of issues:
• rendering improvements including 32-bit and float support
• vast improvements in Mac launching (separate clients for your Java version)
• faster import of complex plates
• OMERO.dropbox improvements
• ROI and measurement tool fixes
• OMERO.matlab updates

3.3.17 5.1.0-m3 (December 2014)

Developer preview release - 3 of 4 development milestones being released in the lead up to 5.1.0. Only intended as a developer preview for updating code before the full public release of 5.1.0. Use at your own risk.

Changes affecting developers include:
• implementation of units in the OMERO clients
• conversions between units
• OMERO.web updates
• server-side Graph work to improve speed for moving and deleting
• OMERO.insight bug-fixes especially for ROIs
3.3.18 5.1.0-m2 (November 2014)

Developer preview release - 2 of 3 development milestones being released in the lead up to 5.1.0. Only intended as a developer preview for updating code before the full public release of 5.1.0. Use at your own risk.

Model changes include:

- units support, meaning units now have real enums
- minor fixes for model changes introduced in m1

The units changes mean that the following fields have changed:

- Plane.PositionX, Y, Z; Plane.DeltaT; Plane.ExposureTime
- Shape.StrokeWidth; Shape.FontSize
- DetectorSettings.Voltage; DetectorSettings.ReadOutRate
- ImagingEnvironment.Temperature; ImagingEnvironment.AirPressure
- LightSourceSettings.Wavelength
- Plate.WellOriginX, Y
- Objective.WorkingDistance
- Pixels.PhysicalSizeX, Y, Z; Pixels.TimeIncrement
- StageLabel.X, Y, Z
- LightSource.Power
- Detector.Voltage
- WellSample.PositionX, Y
- Channel.EmissionWavelength; Channel.PinholeSize; Channel.ExcitationWavelength
- TransmittanceRange.CutOutTolerance; TransmittanceRange.CutInTolerance; TransmittanceRange.CutOut; TransmittanceRange.CutIn
- Laser.RepetitionRate; Laser.Wavelength

Other changes that may affect developers include:

- ongoing C++ implementation improvements
- ongoing work to add unit support in OMERO.insight
- further flake8 work
- removal of webtest app from OMERO.web to a separate repository
- removal of deprecated methods in IContainer and RenderingEngine
- removal of deprecated services IDelete and Gateway
- Blitz gateway fixes
- CLI fixes
- ROI and tables work

3.3.19 5.0.6 (November 2014)

This is a critical security fix for two vulnerabilities:

- 2014-SV3-csrf\(^{27}\)
- 2014-SV4-poodle\(^{28}\)

\(^{27}\)http://www.openmicroscopy.org/info/vulnerabilities/2014-SV3-csrf
\(^{28}\)http://www.openmicroscopy.org/info/vulnerabilities/2014-SV4-poodle
It is strongly suggested that you upgrade your server and follow the steps outlined on the security vulnerability pages. Additionally, a couple of bug fixes for system administrators are included in this release.

### 3.3.20 5.1.0-m1 (October 2014)

Developer preview release - 1 of 3 development milestones being released in the lead up to 5.1.0. **Only intended as a developer preview for updating code before the full public release of 5.1.0. Use at your own risk.**

Model changes include:
- channel value has changed from an int to a float
- acquisitionDate on Image is now optional
- Pixels and WellSample types are no longer annotatable
- the following types are now annotatable: Detector, Dichroic, Filter, Instrument, LightSource, Objective, Shape
- introduction of a “Map” type which permits storing key-value pairs, and a Map annotation type which allows linking a Map on any annotatable object

Other changes that may affect developers include:
- strict flake8’ing of all Python code
- C++ build is now based on CMake and is hopefully much more user-friendly
- new APIs: SendEmail and DiskUsage
- the password table now has a “changed” field

### 3.3.21 5.0.5 / 4.4.12 (September 2014)

This is a critical security fix for two vulnerabilities:
- 2014-SV1-unicode-passwords
- 2014-SV2-empty-passwords

It is highly suggested that you upgrade your server and follow the steps outlined on the security vulnerability pages.

### 3.3.22 5.0.4 (September 2014)

This is a bug-fix release for the Java 8 issues. It also features a fix for uploading masks in OMERO.matlab.

You need to upgrade your OMERO server if you want to take advantage of further improvements in Bio-Formats support for ND2 files.

### 3.3.23 5.0.3 (August 2014)

This is a bug-fix release addressing a number of issues including:
- improved metadata saving in MATLAB
- many bug fixes for ND2 files
- several other bug fixes to formats including LZW, CZI, ScanR, DICOM, InCell 6000
- support for NDPI and Zeiss LSM files larger than 4GB
- export of RGB images in ImageJ
- search improvements
- group owner enhancements

---


• Webclient updates including multi-file download

To take advantage of improvements in Bio-Formats support for ND2 files, you need to upgrade your OMERO.server as well as your clients.

3.3.24 5.0.2 (May 2014)

This is a bug-fix release addressing a number of issues across all components, including:

• import improvements for large image datasets
• shared rendering settings
• better tagging workflows
• disk space usage reporting for OMERO.web admins
• OMERO.matlab annotation handling
• custom Web Start intro page templates
• searching by image ID

To take advantage of improvements in Bio-Formats support for .czi files, you need to upgrade your OMERO.server as well as your clients.

3.3.25 4.4.11 (April 2014)

This is a bug-fix release for the Java Web Start issue. You only need to upgrade if this is a blocker for you and you cannot upgrade to 5.0.x as yet. Also note that the OMERO.insight-ij plugin version 4.4.x no longer works for Fiji, we are working on a fix for this. Plugin version 5.0.x is unaffected.

3.3.26 5.0.1 (April 2014)

This is a bug-fix release addressing a number of issues across all components, including:

• code signing to fix the Java Web Start issues
• stability improvements to search
• MATLAB fixes
• improvements to groups, user menus, file name settings etc
• new import scenario documentation covering ‘in-place’ importing.

3.3.27 5.0.0 (February 2014)

This represents a major change in how the OMERO server handles files at import compared with all previous versions of OMERO. Referred to as ‘OMERO.fs’, this change means that OMERO uses Bio-Formats to read your files directly from the filesystem in their original format, rather than converting them and duplicating the pixel data for storage. In addition, it continues our effort to support new multidimensional images. The changes are especially important for sites working with large multi-GB datasets, e.g. long time lapse, HCS and digital pathology data.

3.3.28 4.4.10 (January 2014)

This is a bug-fix release addressing a number of issues across all components, including:

• improved tile-loading
• better network-disconnect handling
• more flexible
• webapp deployment
• Ice 3.5.1 support (except Windows)
• improved modification of metadata, users and groups

3.3.29 4.4.9 (October 2013)

This is a bug-fix release addressing a number of issues across all components, also including:

• Ice compatibility issues
• new scripting sharing service
• new user help website
• new partner project pages.

The minimum system requirement is Java 1.6 (Java 1.5 is no longer supported).

A security vulnerability was identified and resolved, meaning that we strongly recommend all users upgrade their OMERO clients and servers.

3.3.30 4.4.8p1 (July 2013)

This is a patch release addressing a network connection problem in the clients introduced by a new version of Java.

3.3.31 4.4.8 (May 2013)

This is bug-fix release addressing two specific issues: a problem with the OMERO.insight client for Linux, and image thumbnails not loading for Screens/Plates in Private/Read-Only groups in OMERO.web. You only need to upgrade if you are an OMERO.insight user on Linux or you are using OMERO.web to view HCS data in Private or Read-Only groups.

3.3.32 4.4.7 (April 2013)

This is a point release including several new features and fixes across all components. This includes improvements in viewing of ‘Big’ tiled images, new permission features, new OMERO.web features, and several utility functions in OMERO.matlab.

3.3.33 4.4.6 (February 2013)

This is bug-fix release addressing a number of issues across all components. This includes a major fix to repair the C++ binding support for Ice 3.4. There has also been a potentially breaking update to the CLI.

3.3.34 4.4.5 (November 2012)

This is bug-fix release focusing on improvements to the OMERO clients. OMERO.web now supports “batch de-annotation”, filtering of images by name and improved export to OME-TIFF and JPEG. OMERO.insight has fixes to thumbnail selection and image importing and exporting.

3.3.35 4.4.4 (September 2012)

This is a bug-fix release addressing a number of issues across all components.

• OMERO.insight fixes include connection and configuration options and tagging on import.
• OMERO.web improvements include big image and ROI viewer fixes, improved admin and group functionality and rendering/zooming fixes.
• OMERO.server now has improved LDAP support and VM and homebrew deployments as well as fixes for file downloads above 2GB, permissions, memory leaks and JDK5.
3.3.36 4.4.3 (August 2012)

This is a critical security fix for:

- [2012-SV1-ldap-authentication](http://www.openmicroscopy.org/info/vulnerabilities/2012-SV1-ldap-authentication)

Anyone using OMERO 4.4.2 or earlier with LDAP authentication should immediately upgrade to 4.4.3.

3.3.37 4.4.2 (August 2012)

This release is a major bug fix for archiving files larger than 2 GB. If you do not archive files larger than 2 GB, you do not need to upgrade your clients or your server. There is also a minor fix for an OMERO.imagej plugin security issue, but it is only necessary to update the version of Bio-Formats that is installed in ImageJ.

3.3.38 4.4.1 (July 2012)

This is a minor release which fixes two import issues. See [#9372](https://trac.openmicroscopy.org/ome/ticket/9372) and [#9377](https://trac.openmicroscopy.org/ome/ticket/9377). If you are not using BigTIFF or PerkinElmer .flex files, then you do not need to upgrade.

3.3.39 4.4.0 (July 2012)

This is a major release, which focuses on providing new functionality for controlling access to data, as well as significant improvements in our client applications.

The major theme of 4.4.0 is what we refer to as “Permissions”, the system by which users control access to their data. It is now possible to move data between groups, and much, much more.

We also added a few more things for users in 4.4.0, like:

- OMERO.insight webstart
- Importing from OMERO.insight is now complete
- Better integration of OMERO.insight with ImageJ
- A bottom-to-top reworking of the OMERO.web design

For developers and sysadmins, there are a few things as well:

- Support for Ice 3.4
- Removed support for PostgreSQL 8.3

3.3.40 Beta 4.3.4 (January 2012)

This is a point release is a security update to address an LDAP vulnerability.

3.3.41 Beta 4.3.3 (October 2011)

This point release is a short follow on to 4.3.2 to handle various issues found by users.

3.3.42 Beta 4.3.2 (September 2011)

This is a point release, focusing on fixes for OMERO.web, export, and documentation. A couple of LDAP fixes were also added, following requests from the community. We also included something many of you have asked for some time, OMERO on virtual machines.

---

31)[http://www.openmicroscopy.org/info/vulnerabilities/2012-SV1-ldap-authentication](http://www.openmicroscopy.org/info/vulnerabilities/2012-SV1-ldap-authentication)

32)[https://trac.openmicroscopy.org/ome/ticket/9372](https://trac.openmicroscopy.org/ome/ticket/9372)

33)[https://trac.openmicroscopy.org/ome/ticket/9377](https://trac.openmicroscopy.org/ome/ticket/9377)
3.3.43 Beta 4.3.1 (July 2011)

This point release focuses on fixes for Big Images, OMERO.web and others.

3.3.44 Beta 4.3.0 (June 2011)

This is a major release, focusing on new functionality for large, tiled images, and significant improvements in our client applications.

The major theme of 4.3.0 is what we refer to as “Big Images”, namely images with X,Y images larger that 4k x 4k. With this release, OMERO’s server and Java and web clients support tiling and image pyramids. This means we have the functionality you have probably seen in online map tools, ready for use in any image file format supported by OMERO (and obviously Bio-Formats). This is especially important for digital pathology, and other uses of stitched imaging.

While the major focus of 4.3.0 was Big Images, there are a number of other new updates. For users, we have worked hard to synchronise functionality and appearance across the OMERO clients. This includes viewing of ROIs in OMERO.web. We are not done, but we have made a lot of progress. Moreover, data import is now MUCH faster and available from within OMERO.insight.

3.3.45 Beta 4.2.2 (December 2010)

Fixes blocker reported using 4.2.1. Starting with this milestone, all tickets for the insight client are managed on Trac.

3.3.46 Beta 4.2.1 (November 2010)

This is a point release, focusing on fixes for delete functionality, and significant improvements in the way OMERO.web production server is deployed.

3.3.47 Beta 4.2.0 (July 2010)

This release is a major step for OMERO, enabling a number of critical features for a fully functional data management system:

- User and Group Permissions and data visibility between users
- updates to the OME SPW Model and improvements in HCS data visualisation
- SSL connection between OMERO clients and server;
- full scripting system, accessible from command line and within OMERO.insight, including Figure Export and FLIM Analysis
- ROIs generated in OMERO.insight stored on server
- extended use of OMERO.Tables for analysis results
- performance improvements for import and server-side import histories
- revamped, fully functional OMERO.web web browser interface
- upgrade of Backend libraries in OMERO.server

3.3.48 Beta 4.1.1 (December 2009)

This release fixes a series of small bugs in our previous Beta 4.1 release.

3.3.49 Beta 4.1 (October 2009)

Improved support for metadata, especially for confocal microscopy; OMERO supports all of the file formats enabled by Bio-Formats. Export to OME-TIFF and QuickTime/AVI/MPEG from OMERO. Various improvements to OMERO clients to improve workflow and use.
This release introduces OMERO.qa - a feedback mechanism, to allow us to communicate more effectively with our community. OMERO.qa supports uploading of problematic files, and tracking of responses to any user queries. Moreover, OMERO.qa includes a demo feature: in collaboration with Urban Liebel at Karlsruhe Institute of Technology, we are providing demo accounts for OMERO. Use the Demo link at qa to contact us if you are interested in this.

For users who have had problems with memory-based crashes in OMERO.insight, the new OpenGL-based ImageViewer may be of interest. Also, we are now taking advantage of our modeling of HCS data, and releasing our first clients that support Flex, MIAS, and InCell 1000 file formats. OMERO.dropbox has been substantially extended, and now supports all the file formats supported by OMERO.

### 3.3.50 Beta 4.0.1 (April 2009)

A quick patch release that fixes some bugs and adds some new functionality:

- Fixed Windows installation and updated docs.
- Bug fixes (scriptingEngine, importer).
- Fix .lif import, add Li-Cor 2D (OMERO does gels!).
- API .dv and OME .ome.tif now supported by OMERO.fs.
- Support negative pixel values in Rendering Engine.
- Archived images are now fully supported in OMERO.
- OMERO.web merged with OmeroPy in distribution.

### 3.3.51 Beta 4.0 (March 2009)

This release consists of a major change in the remoting infrastructure, complete migration of existing OMERO clients to the ICE framework, two new OMERO clients, and integration of OMERO.editor into OMERO.insight.

**OMERO.server updates:**

- remove JBOSS, and switch all remoting to ICE
- improve session management, supporting creation of many thousands of session
- addition of an import service for server-side importing
- DB upgrades to support the metadata completion facilities
- substantial improvement to the interaction between the indexing engine and the rest of server.

**OMERO.importer updates:**

- migration to Blitz interface, giving much faster performance
- more efficient importing, complete metadata support for Zeiss LSM510, Leica LIF, Zeiss ZVI, Applied Precision DV, and MetaMorph STK
- addition of command line importer for batch import

**OMERO.insight updates:**

- migration to Blitz interface, giving much faster performance
- updates to metadata display, include complete support for OME Data Model
- much expanded integration of protocol management via OMERO.editor, within OMERO.insight
- support for image delete
- refinement of Projection Interface

**OMERO.web:** all new browser-based client for OMERO. Enables sharing of images with colleagues with an account on server.

**OMERO.editor:** a management tool for experimental protocols, now fully integrated with OMERO.insight, so that protocols and experimental descriptions can be saved along with images and datasets. Includes a new parameters function, so that protocols in traditional documents can be easily imported into OMERO. Supports, tables and .xls files. Also runs as a standalone application.

### 3.3. OMERO version history
OMERO.fs: a new OMERO client, that monitors a specific directory and enables automatic imports. In its first incarnation, has quite limited functionality, supporting automatic import of LSM510 files only.

3.3.52 Beta 3.2 (November 2008)

The final update in the Beta3.x series. A number of fixes:
- faster thumbnailing and better support for large numbers of thumbnails
- improved handling of Leica .lei and Zeiss .zvi files
- extended support for reading OMERO.editor files in OMERO.insight
- measurement tool fixes in OMERO.insight
- fixed memory problem in OMERO.insight on Windows
- fixed thumbnailing and session bugs on OMERO.server
- fixed DB upgrades for older PostgreSQL versions

3.3.53 Beta 3.0 (June 2008)

This release of OMERO is a major update of functionality. In OMERO.server, we have added support for StructuredAnnotations, a flexible data management facility that allows essentially any kind of accessory data to be linked to images and experiments stored in OMERO. Alongside this, we provide an indexing engine, that provides a flexible searching facility for essentially any text stored in an installation of OMERO.server. Finally, we are releasing our first examples of clients that use the OMERO.blitz server, a flexible, distributed interface that supports a range of client environments. One very exciting addition is OMERO matlab, a gateway that can be used to access OMERO from Matlab®.

OMERO Beta3.0 includes a substantial reworking of our clients as well. OMERO.insight has been substantially updated, with an updated interface to provide a more natural workflow and support for many different types of annotations, through the StructuredAnnotations facility. The new search facilities are supported with smart user interfaces, with auto-complete, etc. New file formats have been added to OMERO.importer, including support for OME-XML, and an improved import history facility is now available. Finally, Beta3.0 includes the first release of our experimental electronic notebook tool, OMERO.editor. This represents our recent efforts to capture as much metadata around an experiment as possible.

3.3.54 Beta 2.3.3 insight (April 2008)

A new Beta 2.3.3 OMERO.insight has been released, this adds rotation to ellipse figure, and new format for saving intensity values.

Note: this version saves the ROIs in a format which is incompatible with previous saved ROIs.

3.3.55 Beta 2.3.1 importer (February 2008)

A new Beta 2.3.1 OMERO.importer has been released which includes a number of new formats: Zeiss AxioVision ZVI (Zeiss Vision Image), Nikon NIS-Elements .ND2, Olympus FluoView FV1000, ICS (Image Cytometry Standard), PerkinElmer UltraView, and Jpeg2000.

The OMERO downloads for Beta 2.3 include a number of new options: a new import history feature, a Windows server installation, and a new tagging feature for OMERO.insight.

Note: milestone:3.0-Beta2.3 and prior Mac OS X installers for OMERO.server do not work on Mac OS X Leopard (10.5). Please follow the UNIX-based platform manual install instructions. Mac OS X installers for OMERO.insight and OMERO.importer work just fine under Leopard and can be used.

3.3.56 Beta 2.3 (December 2007)

This is a patch release for OMERO.server to fix a memory problem. In OMERO.insight, updating of the tagging facility, viewing of others’ rendering settings and support for server-side compression of images before transport to client.
3.3.57 Beta 2.2 (November 2007)

In this release we have updated OMERO.server to run a newer version of JBOSS and provided support for copying display settings across a range of images. More new file formats. OMERO.insight has been updated to support copying display settings across many images. Image Viewer has been substantially updated.

3.3.58 Beta 2.1 (August 2007)

This is a client-only release. OMERO.insight now supports basic ROI measurements and a series of new file formats have been added. The OMERO downloads for Beta 2.0 have been simplified. OMERO.insight and OMERO.importer have been combined into a single download file called ‘OMERO.clients’ and the user documentation is now included inside of the server and client downloads.

3.3.59 Beta 2.0 (June 2007)

Note: this version will still work with the Beta 1 server release.

This major update provided our first support for multiple platforms via OMERO.Blitz. OMERO.insight now supports viewing work of multiple users. Beta 2 is our first release of the Web2.0-like ‘tag’ system developed in collaboration with Usable Image34. This version addresses issues with using our tools under Java 1.6

3.3.60 Beta 1.1 (March 2007)

Patch release to fix time-out issues.

3.3.61 Beta 1 (January 2007)

The first public OMERO release, providing simple data management. Limited file format support (DV, STK, TIFF). Simple data visualization and management.

3.3.62 Milestone M3 (November 2006)

Rendering and compression API and client-side import. Access control and permissions system. Importer based on Bio-Formats.

3.3.63 Milestone M2 (July 2006)

The stateful rendering service is functional and all rendering code moved from Shoola Java client to the server. Also, the stateless services (IQuery, IUpdate, IPojos) are frozen and testing and documentation is checked and solidified.

3.3.64 Milestone M1 (April 2006)

Contains minimal functionality needed to run Shoola Java client without Perl server to demonstrate acceleration of metadata access. Application deployed on JBoss (http://www.jboss.org). No ACLs or permissions.

34http://www.openmicroscopy.org/site/about/project-history/catriona
QUICKSTART SERVER ACCESS

If your institution does not have an existing OMERO.server for you to connect to, you can apply for an account on our demo server. Alternatively, you can create your own using a virtual appliance (a step-by-step guide for how to do this is provided).

Note: The virtual appliance is being discontinued. Previous versions will remain available for download but OMERO 5.2.1 will be the most recent release.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

4.1 OMERO demo server

Take advantage of the OMERO platform and ask for a free demo account.

To register for a free demo account, please visit the registration page\(^1\).

We will contact you and provide you with a server name, username and password, please be patient as it may take a few days to process your application after we receive it.

If you have already requested a demonstration account:

- your username and password will be emailed to you.

Once you have these, follow the instructions below.

4.1.1 Service-level agreement

By using the OMERO Demo Server, you agree to the following:

- You use this service at your own risk, and we provide NO GUARANTEE OR WARRANTY WHATSOEVER regarding uptime, security, data provenance or access, or any other service.
- You should not use the OMERO Demo server as a repository for any important data, or data that must be held in any secure or specific way.
- The responsibility for any data loaded onto the OMERO Demo Server is entirely your own.
- We reserve the right to remove any images, data, accounts or other data or structure without notice or request.

We welcome comments or ideas from your use of and experience with the OMERO Demo Server on the OMERO User discussion forum\(^2\) or ome-users mailing list\(^3\).

4.1.2 Getting started with your demo account

To log on and try the demo server you have two options:

\(^1\)http://qa.openmicroscopy.org.uk/registry/demo_account/
\(^2\)http://www.openmicroscopy.org/community/viewforum.php?f=4
\(^3\)http://lists.openmicroscopy.org.uk/mailman/listinfo/ome-users/
1. **Using the OMERO.insight client**

Download the OMERO.insight client for your platform, install it and run it as a stand-alone application. You will need to have permissions allowing you to install software on your computer, and you will need to update the client if the demo server is upgraded (we will email you to inform you).

2. **Using the OMERO.web client**

The web client does not allow you to import data or create new ROIs.

---

**Using the OMERO.insight client**

Download the OMERO.insight client for your platform by clicking on the appropriate link on the [downloads page](http://downloads.openmicroscopy.org/latest/omero/).

After download, unzip and install the OMERO.insight client as you would for any other application.

Launch OMERO.insight, and you will see the login screen.

Click the **Spanner** icon.

In the **Servers** window, click the + icon.

---

4http://downloads.openmicroscopy.org/latest/omero/
Enter the **Server Address**: demo.openmicroscopy.org
The port will automatically be set to 4064.
Select your network connection speed.
Click **Apply**

Enter the username and password provided.
Click **Login**

Using the OMERO.web client

To use the OMERO.web client click on the following link:
**OMERO.web**

The server name is entered automatically, and you enter your demo account username and password as described above.
Please change your password as soon as you have logged in. The easiest way to do this is via: **Web Admin**

---

5 [http://demo.openmicroscopy.org/webclient/login](http://demo.openmicroscopy.org/webclient/login)
4.1.3 Further guides and help

- Workflow-based user assistance guides are available on our help website. These use screenshots to illustrate how to perform common tasks such as importing and viewing data, and using the measurement tool.
- An overview of some OMERO’s features is available in a short showcase video at: Showcase Video
- If you have any questions, suggestions, comments or proposals, please take advantage of the OME community’s expertise and join our forums or mailing lists:
  - Forums
  - Mailing Lists
- For more information on OME and OMERO please visit:
  - Open Microscopy Environment

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

4.2 OMERO virtual appliance

Note: The virtual appliance is being discontinued. Previous versions will remain available for download but OMERO 5.2.1 will be the most recent release.

The OMERO virtual appliance is a quick, easy, and low-cost way to try out OMERO.server on your laptop or desktop. This enables you to make an informed decision about whether committing to an OMERO.server install is right for you.

Virtualization enables canned, ready to run software environments to be created and used, in the form of VM (Virtual Machine), or to be distributed for others to use, in the form of virtual appliances. A Virtual Appliance is essentially a file that describes how to create a new Virtual Machine on demand. The virtualized software environment can contain an entire OS (Operating System), such as Windows or Linux, and any other software that runs in that OS, such as, in this case, OMERO.server and its associated software prerequisites. Once created and started, you can log into the OS and use it as though it were a real machine. One way to think of this is as though you had an entire computer in a window on your desktop.

When using virtualization software, the OS that is running the virtualization software is referred to as the “host OS”. When you use virtualization, the OS running within a virtual machine is referred to as the “guest OS”. This allows us to be explicit about which OS we are working in.

This technology allows the OME Project to distribute a canned, ready-to-run environment containing an OMERO.server, freeing you from having to install the server and prerequisites yourself, and letting you concentrate on evaluating the functionality of the OMERO platform.

Note: The virtual hard-drive used by the OMERO virtual appliance is 30GB in size and you should keep track of the amount of this space you have consumed and, if necessary, delete data that is not required. If your data is likely to exceed this space whilst you are evaluating OMERO then it is worthwhile going through the Increasing HD size before you start working with OMERO in earnest.

4.2.1 Getting started

To use the virtual appliance you should do the following:

- Install VirtualBox
• Download the OMERO.server virtual appliance
• Import the virtual appliance into VirtualBox to create a virtual machine
• Start the virtual machine

Each of these points is outlined in full detail below. Once you have everything installed, the virtual appliance should just work straight out of the box but if you have any problems, there is further guidance available for Troubleshooting OMERO virtual appliance and Working with OMERO virtual appliance.

Install VirtualBox

Download VirtualBox from the VirtualBox Downloads page and follow the installation process for your platform. If in doubt, you should download, or upgrade to, the latest version of VirtualBox. Once VirtualBox is installed, run the application. Depending upon your platform and version, the VirtualBox interface should look similar to the following screenshot:

![VirtualBox installation](image)

Figure 4.1: VirtualBox installation

Download the OMERO.server virtual appliance

The virtual appliance has its own download page and will have a filename similar to omero-va-5.2.6.ova.

Import OMERO virtual appliance into VirtualBox

• Start VirtualBox then select ‘File/Import Appliance’. You will be presented with a dialog box.
• Select and navigate to the location where you downloaded the virtual appliance file.
• Select your OVA file then click open.

This process is indicated in the screenshot below.

• Click continue. You will be presented with a range of options for the VM that will be built from the appliance.
• You can accept the defaults by clicking Import.

13 http://downloads.openmicroscopy.org/latest/omero-virtual-appliance/
14 https://www.virtualbox.org/wiki/Downloads
15 http://downloads.openmicroscopy.org/latest/omero-virtual-appliance/
You should now see a progress bar as your new virtual machine is built from the appliance. This may take a few minutes depending upon your hardware.

When the import procedure is complete, your new VM should appear in the VirtualBox VM library ready for use.

**Networking**

Our virtual appliance is distributed with VirtualBox’s built in Host-Only Network Address Translation (NAT) preconfigured. This means that the IP address for the VM is 10.0.2.15 as this is the default VirtualBox Host-Only NAT address. Using this address is the simplest way to distribute a virtual appliance when you do not know the setup of a user’s network.

**Port-forwarding settings**

Your host OS cannot connect directly to 10.0.2.15 but needs to use port-forwarding. This means that you connect to your localhost on a specific port and the communications to and from that port are forwarded to specified ports on the guest VM.
Our virtual appliance should be preconfigured with the correct port-forwarding setting during the import process. However, it is best to double check that these settings are correct:

- Select your VM in the VirtualBox VM Library
- Click on Settings then select the Network tab
- Click on Advanced
- Click on Port Forwarding

If the table in the window that appears is empty then port forwarding is not setup. The required port-forwarding settings are as follows:

<table>
<thead>
<tr>
<th>Name</th>
<th>Protocol</th>
<th>Host IP</th>
<th>Host Port</th>
<th>Guest IP</th>
<th>Guest Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>omero-ssl</td>
<td>TCP</td>
<td>127.0.0.1</td>
<td>4064</td>
<td>10.0.2.15</td>
<td>4064</td>
</tr>
<tr>
<td>omero-unsec</td>
<td>TCP</td>
<td>127.0.0.1</td>
<td>4063</td>
<td>10.0.2.15</td>
<td>4063</td>
</tr>
<tr>
<td>omero-web</td>
<td>TCP</td>
<td>127.0.0.1</td>
<td>8080</td>
<td>10.0.2.15</td>
<td>8080</td>
</tr>
<tr>
<td>ssh</td>
<td>TCP</td>
<td>127.0.0.1</td>
<td>2222</td>
<td>10.0.2.15</td>
<td>22</td>
</tr>
</tbody>
</table>

When correctly setup in VirtualBox, your port forwarding settings should look like this:

If you are on Linux or Mac OS X, you can either use our port forwarding setup script or you can set up port forwarding manually. On Microsoft Windows systems you will have to set up port forwarding manually as the script requires a Bash shell.

The script can be downloaded from the online version of this documentation; see [http://www.openmicroscopy.org/site/support/omero/users/virtual-appliance.html](http://www.openmicroscopy.org/site/support/omero/users/virtual-appliance.html).

After obtaining the script, it can be used in the following manner:

```bash
$ bash setup_port_forwarding.sh $VMNAME
```

where $VMNAME is the name of your VM.

**Note:** By default the scripts create a VM named omerovm and the pre-built appliance is named omero-vm

Adding port forwarding manually is achieved by editing the port forwarding table shown above. Use the + to add a new row to the table, then click in each cell and type in the required settings.

Now you are ready to start your VM. Select the VM in the VirtualBox VM library then click **start**.
A window should open containing a console for your VM which should now be going through its standard boot process. OMERO.server is automatically started at boot time, meaning that you should be able to interact with OMERO without further setup.

**Credentials**

The following accounts are preconfigured in the OMERO virtual appliance: an OS account, for logging into the VM as the `omero` user, and a single OMERO.server account which is used to access the OMERO.server software as the OMERO.server `root` user.
Virtual Appliance OS credentials

<table>
<thead>
<tr>
<th>Username</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>omero</td>
<td>omero</td>
</tr>
</tbody>
</table>

A ‘root’ OS account is not needed as the omero account is able to run administrative commands via sudo with no password required.

OMERO.server credentials

<table>
<thead>
<tr>
<th>Username</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>omero</td>
</tr>
</tbody>
</table>

You can use this administrative account to create as many user level accounts as you require in the usual way.

4.2.2 Using the virtual appliance

Further guidance on how to interact with the server and troubleshoot common issues is available. See Working with OMERO virtual appliance and Troubleshooting OMERO virtual appliance. In particular, increasing the base memory allocation may significantly improve import performance.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

Working with OMERO virtual appliance

Now that your VM is up and running you have a choice about how to interact with it.

- You can connect to OMERO.web from your host browser. Go to http://localhost:8080/webclient.
- You can use OMERO.clients from within your host OS. This will allow you to import data via a GUI and manage that data once imported. To do so, download the OMERO.insight client and follow the instructions below. More information can be found on our help website which provides workflow-based guides to using the OMERO.clients.
- Alternatively, you can interact with the server command line interface by SSH (Secure Shell)’ing into the guest VM or by opening a console within the VM itself. Administrators may need to use one of these methods to restart the server and/or change configuration parameters. In this case, you must have an SSH client installed on your host machine to use to connect to the OMERO.server.

Note: The following examples assume that the OMERO VM is up and running on the same machine that you are working on.

OMERO.web

Go directly to http://localhost:8080/webclient to log in with user: “root” / pw: “omero”.

Note: If you receive a 502 nginx error on first attempting to connect to the web app on http://localhost:8080/webclient/ please restart the virtual machine and try again.

OMERO.insight

You can run regular OMERO clients on your host machine and connect to the server in the VM. Our example uses OMERO.insight running on Mac OS X to connect to the VM.

- Download and install OMERO.insight

16 http://downloads.openmicroscopy.org/latest/omero/
17 http://help.openmicroscopy.org/
18 http://downloads.openmicroscopy.org/latest/omero/
• Start OMERO.insight
• Click the spanner icon situated above the password box to enter the server settings box shown below.

![Server Settings](image)

Figure 4.7: Setting OMERO.insight server address and port number

• Use the + icon to add a new server entry with the address localhost and the port 4064 then click apply
• You can now use the login credentials given above to log into OMERO.insight using the login window shown below (user: “root” / pw: “omero”).

![Login Window](image)

Figure 4.8: OMERO.insight login window

• OMERO.insight should now load up and display the main window.
You can now use OMERO.insight to import and manage images on a locally running virtual server just like you would use the standard remote server.

**Note:** A Getting Started guide is available for OMERO.insight on our help website[^19] if you need further assistance to download and install the software.

### Secure Shell

You can log into your VM using SSH, allowing you to use the **Command Line Interface as an OMERO client**. In the following example, we assume that you have an SSH client installed on your host machine and also that your VM is up and running.

You can log into the VM using the above credentials and the following command typed into a terminal:

```
$ ssh omero@localhost -p 2222
```

This invokes the SSH program telling it to login to the localhost on port 2222 using the username omero. Remember that earlier you set up port forwarding to forward port 2222 on the host machine to port 22 (the default SSH port) on the guest VM. You should be prompted for a password. Once you have successfully entered your password, you should be greeted by a prompt similar to the following:

```
omero@omerovm:~$
```

There are two potential complications to this method, (1) if you have used a VM before then there could be old SSH fingerprints set up, (2) the first time that you log into the VM you will be asked to confirm that you wish to continue connecting. If you get the following message after you invoke SSH, you need to remove the old fingerprints:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
Please contact your system administrator.
Add correct host key in /Users/rleigh/.ssh/known_hosts to get rid of this message.
Offending key in /Users/rleigh/.ssh/known_hosts:14
RSA host key for localhost has changed and you have requested strict checking.
Host key verification failed.
```

You can do this using the following command typed into the terminal:

```
$ ssh-keygen -R [localhost]:2222 -f ~/.ssh/known_hosts
```

This should produce output similar to:

```
$ ssh-keygen -R [localhost]:2222 -f ~/.ssh/known_hosts
/Users/rleigh/.ssh/known_hosts updated.
Original contents retained as /Users/rleigh/.ssh/known_hosts.old
```

The first time that you log into the VM you will also be asked to confirm that you wish to connect to this machine by a message similar to the following:

```
$ ssh omero@localhost -p 2222
The authenticity of host '[localhost]:2222 ([127.0.0.1]:2222)' can't be established.
Are you sure you want to continue connecting (yes/no)?
```

You should confirm that you wish to continue connecting, after which you will be prompted for your password as usual:

```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[localhost]:2222' (RSA) to the list of known hosts.
omero@localhost's password:
```

After which, you should have a prompt indicating that you have a shell open and logged into the VM:
Log into the VM directly

**Note:** Due to the frequent changes in the VirtualBox Guest Additions, key mappings between the host and guest OS do not always work. We recommend using SSH as the primary way of interacting with the Virtual Appliance CLI.

When you start your VM using the VirtualBox GUI, as outlined above, a window will be displayed showing the boot process for the machine as it starts up, just like with a real piece of hardware. Once the boot process has finished you will see a prompt displayed in this window like so:

```
[System startup messages]
Debian GNU/Linux 6.0 omerovm tty1
omerovm login: _
```

You can log into the console of the VM directly using the user account credentials above.

```
omerovm login: omero
Password: _
```

There is no GUI on the current OMERO virtual appliance so you will have to use the Bash shell which looks like this:

```
omerovm:~$ _
```

From here you can interact with OMERO.server via the *Command Line Interface as an OMERO client*. You will need to login as the ‘omerou’ user to access the OMERO CLI (user: “omerou” / pw: “omero”). Logout using Ctrl-D.

**Note:** This documentation is for **OMERO 5.2**. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

Troubleshooting OMERO virtual appliance

**Networking not working**

Occasionally, during the boot process, the VirtualBox DHCP server fails to allocate an IP address to the OS in the guest VM. This means that OMERO.clients, such as OMERO.insight, cannot connect to the OMERO.server in the VM.

**CAUSE:** We believe that this is an intermittent VirtualBox bug that resurfaces across many versions VirtualBox #4038\(^20\) and previously VirtualBox #3655\(^21\).

\(^20\)https://www.virtualbox.org/ticket/4038
\(^21\)https://www.virtualbox.org/ticket/3655

4.2. OMERO virtual appliance
• **DIAGNOSIS:** Check whether the guest VM has been allocated the reserved host-only NAT IP address. If 10.0.2.15 does not appear in the output from `ifconfig` then this issue has occurred. The easiest way to verify this is to log into the guest VM console and check the output from executing the following command:

\ `$ \text{ifconfig}$`

• **SOLUTION:** An easy, but unreliable, fix is to reboot the guest VM. The preferred fix is to log into the guest VM console and execute the following commands, which will cause the guest OS to release its IP lease before requesting a new lease:

\ `$ \text{dhclient -r}$
\ `$ \text{dhclient -eth0}$`

**Port conflict when OMERO.server already running in Host OS**

If you are already running an instance of the OMERO.server in your host OS then there will be a conflict due to the ports assigned to VirtualBox port-forwarding already being in use.

• **SOLUTION 1:** Turn off the OMERO.server in the host environment by issuing the following command:

\ `$ \text{omero admin stop}$`

• **SOLUTION 2:** Alter the port-forwarding settings for your OMERO.VM as described in the Port-forwarding settings section. For example, increment the host port settings for omero-ssl, omero-unsec, and omero-web.

**Note:** We are assuming that your host OS is not already running services on those ports. You can check whether something is already listening on any of these ports by running the following commands (Mac OS X) which should return the prompt without any further output if there is nothing listening:

\ `$ \text{lsot -nP | grep -E \'^(:4063)|(:4064)\'$}$`

**Slow import speed due to low memory allocation**

By default, the base memory of the VM is set to 1024 MB to allow it to run on any machine. However, this is likely to result in very poor import performance such that increasing the memory allocation to 2048 MB should result in import speeds being improved by a factor of 3-4 times.

To increase the base memory allocation, right-click on the OMERO virtual appliance in Virtual Box and select Settings:

Then go to the System tab:

Increase the Base Memory by typing into the text box:

If your VM is already running, you will need to restart it for this change to take effect.

Alternatively, you can use the command line:

\ `$ \text{VBoxManage modifyvm OMERO.vm-5.2.6-bxx --memory 2048}$`

**VM will not boot because the HDD (Hard Disk Drive) is full**

If you fill the virtual HDD used by your VM then the OS may be unable to boot and you will lose access to your OMERO.server install. You may also get a “error 28: no space left on device” message. To log into your VM you will need to use the recovery mode. Start the VM and at the Grub screen, use the down arrow followed by return to select the recovery mode entry, e.g.
Debian GNU/Linux, with Linux 2.6.32-5-686 (recovery mode)

as illustrated in this example of the Grub screen:
Do not worry if your VM has a kernel number different to 2.6.32-5-686, the important thing is that you select the entry labeled “recovery mode”. At this point, the VM should rapidly boot into the recovery mode and enable you to log in as the `omero` user. Administrative commands may be run as root by using `sudo`.

Once you have logged in, you have a number of options but the recommended courses of action are:

1. Delete unnecessary files using standard Linux command line tools like `rm` to make space for the VM to boot normally, then use your favored OMERO client to login and delete more files. A useful place to start might be by deleting the logs stored in `/var/logs`.

2. Increase the size of your virtual HDD. If you have filled your existing HDD then it is likely that the volume of data you are storing is too big for the default HDD. You should use the instructions in the `Increasing HD size` section following to ensure that the size of virtual HDD you have available is suitable for the volumes of data that you are collecting.

### Increasing HD size

Image data can become very large and easily fill available hard-drive space. By default, the OMERO virtual appliance is supplied with a 30GB virtual hard-drive. Before using the appliance, consider the volume of data you will be working with whilst evaluating OMERO and whether you need to increase the size of the virtual hard-drive to accommodate it.

The following is a step-by-step guide; be aware that this is not a risk-free procedure and you should backup your VM before proceeding.

#### Preliminary steps

**Acquiring a Ubuntu Linux ISO**  Download an [Ubuntu Linux ISO](http://www.ubuntu.com/download/desktop). The most up-to-date version is fine.

**Back up your VM**  Before you proceed further, you should create a clone of the omero-vm and subsequently work on the copy so that if something gets broken you can always start again. The easiest way to do this is from the command line.

**Note:** If you are on Windows then you should navigate to `C:\Program Files\Oracle\VirtualBox\` because the VBoxManage tools are not added to your path by default.

Start a shell and, assuming that your VM has the default name of omero-vm, use the following command:

**Endnote:**

22http://www.ubuntu.com/download/desktop
$ VBoxManage clonevm omero-vm --mode machine --options keepallmacs --name omero-vm-2 --register

This will create a copy of your VM called omero-vm-2 which you can make alterations to. This means that you can always return to the original omero-vm if you break anything. From now on only make changes to omero-vm-2.

**Extending the HDD**

By default, your virtual hard-drive attached to omero-vm-2 is of a type which cannot be extended; so you need to change this by cloning your HDD from the VDMK type to VDI type:

$ VBoxManage clonehd omero-vm-2-disk1.vmdk omero-vm-2-disk1.vdi --format VDI

You now need to increase the size of your virtual HDD. The following command resizes the HDD to 60GB but you should select a size to suit the amount of data you plan to store in OMERO:

$ VBoxManage modifyhd omero-vm-2-disk1.vdi --resize 60000

**Adding the extended HDD to the VM clone** You now need to tell VirtualBox to use omero-vm-2-disk1.vdi instead of omero-vm-2-disk1.vmdk which is currently attached to the VM. Whilst you are on the Storage tab you will also attach the Ubuntu ISO that you downloaded earlier to your VM. This will allow you to use the tools that ship with Ubuntu to make changes to the filesystem within your VM.

1. Start VirtualBox and select omero-vm-2 in the VM library.
2. Right-click Settings then select the Storage tab.
3. Right-click on omero-vm-2-disk1.vmdk and select Remove attachment.
4. Next to the SATA Controller entry, click the Add Hard Disk icon with a green plus sign. In the pop-up dialog, select Choose existing disk. Now navigate to the location where VirtualBox stores your virtual machines and enter the omero-vm-2 directory. Select the omero-vm-2 disk1.vdi and click open.
5. Add an IDE Controller using the Add Controller icon. Select this new controller then click Add CD/DVD device followed by Choose Disk. Navigate to the location of your Ubuntu ISO, select it and click OK.

The storage for your OMERO VM should now look similar to Virtual Appliance storage settings. Click OK to return to the VirtualBox VM library. With omero-vm-2 selected, ensure that the storage details match what you expect, e.g. omero-vm-2-disk1.vdi is connected to your SATA Port 0. The size for this disk should also more or less match what you specified earlier with the VBoxManage modifyhd command. The reported numbers do not exactly matchup, e.g. a virtualised HDD of 60GB size will be reported as 58.59GB.

**Reallocation space on the VA HDD** Start the omero-vm-2 VM. Ubuntu Linux should boot and you should eventually see a welcome screen giving you the option to try Ubuntu or to install it. You can now start the GParted software and resize your partitions.

1. Select try Ubuntu and you should be presented with a graphical desktop.
2. Start the gparted tool using the menu option under System → Administration → GParted Partition Editor.
3. The GParted GUI will display information similar to Virtual Appliance HDD in GParted.
4. Right-click the entry for /dev/sda5 and select Swapoff.
5. Right click on /dev/sda5 and click Delete to remove the swap partition.
6. Delete /dev/sda2 in the same way. This should leave two entries, one for /dev/sda1 and one for unallocated space.
7. Right-click /dev/sda1 and select Resize. Now drag the right arrow to the right until the entry for Free space following (MiB) is about 2000, then click Resize/Move.

4.2. OMERO virtual appliance
8. Right click the entry for unallocated space and select *New* from the pop-up menu. Select *linux-swap* from the *File system* drop-down menu then *Add*.

![Figure 4.9: Virtual Appliance storage settings](image)

Up until this point you have not actually applied any of your changes to the HDD, you have only specified a list of changes that should be made. You can now go ahead and apply them by selecting the *Edit → Apply All Operations* menu item, then clicking *Apply* in the confirmation dialog box.

When the operations have completed, dismiss the dialog with the *Close* button, close GParted, then shutdown the VM.

**Changing HDD settings inside the VA**  You no longer need the Ubuntu ISO so you can detach it from your VM. Ensure that omero-vm-2 is selected then click *Settings* and select the *Storage* tab. Right-click the *IDE Controller* entry and select *Remove Controller*, then click *OK* to return the VirtualBox VM library.

![Figure 4.10: Virtual Appliance HDD in GParted](image)
Start the omero-vm-2 VM and allow it to boot. As root then issue the `df -h` command. Verify that the size of the `/dev/sda1` is approximately what you expect, e.g. if you allocated a 60GB virtual HDD then after size conversions and swap allocation you should end up with `/dev/sda1` reported as being around 56GB.

Within the VM you need to add the UUID of the new swap partition to the `/etc/fstab` file because you deleted the old one and created a new swap meaning that the IDs will no longer match.

```
$ vim /etc/fstab
```

Move your cursor to the entry that looks similar to the following:

```
UUID=SOME-LONG-ALPHA-NUMERIC-STRING none swap sw 0 0
```

then press `i` to enter “insert mode”. Delete the alphanumeric string so that the entry looks similar to the following:

```
UUID= none swap sw 0 0
```

and place your cursor after the equals sign. You can now issue a command from within the VIM editor to insert your new swap UUID into the `fstab` file.

```
[Insert Mode] <CTRL-R> =system('/sbin/blkid -t TYPE=swap | cut -c18-53') <return>
```

Save your file and quit VIM:

```
[Command Mode] :wq <return>
```

Now reboot your VM with

```
$ shutdown -r now
```

Once your VM has rebooted you should now have a working VM with a larger virtual HDD.

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.
Part II

System Administrator Documentation
5.1 What’s new for OMERO 5.2 for sysadmins

- OMERO clients are no longer distributed as Java Web Start applications. This decision is discussed at length in our Java Web Start blog post. We recognize that this will be problematic for some institutional users and are working to expand the functionality of OMERO.web to mitigate this as far as possible.

- The Version requirements section provides extensive details about which operating systems and dependency versions we intend to support for the life of 5.2 and the likely changes to these for the next major release (currently planned to be 5.3). Most notably, we have dropped support for Java 1.6 and changed the Python dependencies.

- The OMERO.web framework no longer bundles a copy of the Django package, instead manual installation of the Django dependency is required. It is highly recommended to use Django 1.8 (LTS) which requires Python 2.7. For more information see Python on the Version requirements page.

- FastCGI support was removed in OMERO 5.2 and OMERO.web can be deployed using WSGI Deployment (Unix/Linux).

- The configuration property omero.graphs.wrap which allowed switching back to the old server code for moving and deleting data has now been removed. You must migrate to using the new request operations.

- Add support for Ice 3.6. With Ice 3.6, the Python bindings are provided separately. This allows to install the RPM packages provided by ZeroC. Then run `pip install zeroc-ice` to install the Ice Python bindings if your package manager does not provide the Ice python packages.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

5.2 Server overview

The OMERO server system provides storage and processing of image data which conforms to the OME Specification. It can be run on commodity hardware to provide your own storage needs, or run site-wide to provide a large-scale collaborative environment.

Although getting started with the server is relatively straightforward, it does require installing several software systems, and more advanced usage including backups and integrated logins, needs a knowledgeable system administrator.

You may find the OMERO clients overview user guide useful before working through the installation and maintenance guides provided in this section of the documentation.

2https://docs.djangoproject.com/en/1.8/releases/1.8/
3http://www.openmicroscopy.org/site/support/ome-model/specifications/
5.2.1 Developing the server

The server system is composed of several components, each of which runs in a separate process but is co-ordinated centrally.

- **OMERO.blitz** - the data server provides access to metadata stored in a relational database as well as the binary image data on disk.

- **OMERO.dropbox** - a filesystem watcher which notifies the server of newly uploaded or modified files and runs a fully automatic import (designed as the first implementation of OMERO.fs referred to in the architecture diagram).

If you are interested in building components for the server, modifying an existing component, or just looking for more background information, there is a section about the server within the Developer Documentation; the best starting point is the OMERO.server overview for developers.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

5.3 System requirements

5.3.1 Hardware

**OMERO.server**

The system requirements for OMERO.server vary greatly depending on image size and number of users. At a minimum we suggest:
• Single core 1.33GHz Intel or AMD CPU
• 2GB RAM
• 500MB of hard drive space for OMERO.server distribution

The recommended OMERO.server specification we suggest for between 25-50 users is:
• Quad core 1.33GHz Intel or AMD CPU
• 8GB RAM
• 500MB hard drive space for OMERO.server distribution
• Hard drive space proportional to the image sizes expected (likely between 10 and 100TB)

Hard drive space should be proportional to the image sizes expected. The drive space should permit **proper locking**, which is often not the case with remotely mounted shares. See the *Unix* and *Windows* binary repository sections for more information.

RAM is not going to scale linearly, particularly with the way the JVM works. You are probably going to hit a hard ceiling between 4 and 6GB for JVM size (there is really not much point in having it larger anyway). With a large database and aggressive PostgreSQL caching your RAM usage could be larger. Still, even for a large deployment, it is not cost effective to use more than a few GBs of RAM for this purpose. *Performance and monitoring* provides information about fine-tuning the server processes’ memory usage. In summary, depending on hardware layout 16, 24 or 32GB of RAM would be ideal for your OMERO server. If you have a separate database server more than 16GB of RAM may not be of much benefit to you at all.

CPU is not something that an OMERO system is usually ever limited by. However, when it is limited, it is almost always limited by GHz and not by the CPU count. So you are not going to get a huge OMERO performance increase by, for example, throwing 24 cores at the problem. In summary, depending on hardware layout 2 × 4, 2 × 6 system core count should be more than enough.

*Example production server set-ups* provides details on some production set-ups currently in use by OMERO admins, along with how many users and the amount of data they support, which you may find helpful.

**OMERO.insight and OMERO.importer**

The recommended client specification is:
• Single core 1.33GHz Intel or AMD CPU
• 2GB RAM
• 200MB hard drive space for OMERO.clients distribution

Large imports may require 4GB RAM.

**Client configuration**

When performing some operations the clients make use of temporary file storage and log directories. The table below indicates the default values for each directory and the environment variables for overriding their locations:

<table>
<thead>
<tr>
<th>Client directory</th>
<th>Environment variable</th>
<th>Default location (UNIX)</th>
<th>Default location (Windows)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMERO.user directory</td>
<td>OMERO_USERDIR</td>
<td>$HOME/omero</td>
<td>%HOMEPATH\omero</td>
</tr>
<tr>
<td>Temporary files</td>
<td>OMERO_TMPDIR</td>
<td>$HOME/omero/temp</td>
<td>%HOMEPATH\omero\tmp</td>
</tr>
<tr>
<td>Local sessions</td>
<td>OMERO_SESSIONDIR</td>
<td>$HOME/omero/sessions</td>
<td>%HOMEPATH\omero\sessions</td>
</tr>
<tr>
<td>Log files</td>
<td></td>
<td>$HOME/omero/log</td>
<td>%HOMEPATH\omero\log</td>
</tr>
</tbody>
</table>

Note that setting **OMERO_USERDIR** will also change the default location for the temporary files and the local sessions.

If your home directory is stored on a network, possibly NFS mounted (or similar), then these temporary files are being written and read over the network. This can slow access down.

**See also:**

*Troubleshooting performance issues with the clients* Troubleshooting section about client performance issues on NFS
5.3.2 Software

Each component of the OMERO platform has a separate set of prerequisites. Where possible, we provide tips on getting started with each of these technologies, but we can only provide free support within limits.

<table>
<thead>
<tr>
<th>Package</th>
<th>OMERO.server</th>
<th>Java</th>
<th>Python</th>
<th>Ice</th>
<th>PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMERO.importer</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td>OMERO.insight</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td>OMERO.server</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td>OMERO.web</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td>OMERO.py</td>
<td>Required for some functionality</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
<td></td>
</tr>
<tr>
<td>OMERO.cpp</td>
<td>Required for some functionality</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
<td></td>
</tr>
</tbody>
</table>

For full details on which versions of these are supported for OMERO 5.2 and how we intend to update these going forward, see the Version requirements section.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

5.4 Version requirements

5.4.1 Summary of changes for OMERO 5.2 and planned changes for 5.3

Criteria for what is considered to be supportable includes whether support by both upstream developers and operating system distributions will be available for the lifetime of the 5.2 release (including security support), and also upon our resources allocated to CI and testing. If we are not actively testing it, we cannot claim it is supported or functional. Software components must be provided and supported either by an operating system distribution or their original developers.

This section contains a summary of the changes made to the minimum version requirements for the 5.2 release and also planned changes for the following 5.3 release, albeit tentatively at this point. The intent is to provide a roadmap in order that sysadmins may plan ahead and ensure that prerequisites are in place ahead of time to ease future upgrades. The following sections provide more detailed information and the rationale for the changes.

Operating systems

- **Microsoft Windows**
  - [5.2] 7 / Server 2008R2 supported, 8 (client) recommended / Server 2012R2 (server) recommended
  - [5.3] Unsupported for OMERO.server or hosting OMERO.web, client support only
  - **Rationale:** see blog post explanation

- **MacOS X**
  - [5.2] 10.8 dropped. 10.9 deprecated, 10.10 recommended, 10.11 upcoming
  - [5.3] 10.9 dropped, 10.10 deprecated, 10.11 recommended, 10.12 upcoming
  - Test server only on 10.10+ for 5.2.
  - **Rationale:** 10.9 no longer has security support and it still leaves 3 versions to support. MacOS X is typically suited only to client use, not serious server deployment, so limit testing accordingly.

- **Linux (CentOS/RHEL)**
  - [5.2] 7.x recommended, 6.x supported
  - [5.3] 7.x recommended, 6.x deprecated
  - **Rationale:** There is still a significant CentOS 6 userbase, so retain support for 5.2 and potentially continue to support for 5.3 depending upon demand and usage.

- **Linux (Ubuntu LTS)**

---

4http://blog.openmicroscopy.org/tech-issues/future-plans/deployment/2016/03/22/windows-support/
OMERO Documentation, Release 5.2.6

• [5.2] 12.04 dropped, 14.04 recommended
• [5.3] 12.04 dropped, 14.04 recommended
• Only LTS versions are supported and tested. Non-LTS versions following a supported LTS release are likely to work but are not officially supported or tested.
  • Rationale: No change in this timeframe.

Linux (Debian)
• [5.2] 7 deprecated, 8 recommended
• [5.3] 7 deprecated, 8 recommended
• Rationale: No change in this timeframe.

Bitness
Currently 32- and 64-bit systems are supported for client and server on all platforms with the exception of MacOS X (64-bit only).
• [5.2] Dropped 32-bit support for Ice and native code [server]
• [5.3] Deprecate 32-bit support for Ice and native code [client]
• Rationale: It is not practical to run a server on a 32-bit system due to its memory requirements, so it is safe to drop 32-bit support for OMERO.server in 5.2 given that it is de-facto 64-bit-only today. There are still a significant number of 32-bit Windows clients (~60%), so support should be retained until this changes. Maintaining custom builds of Ice for 32-bit systems is extremely costly, and so we could consider dropping them for 5.3. This also depends upon Ice support for 32-bit systems.

Components
• PostgreSQL
  • [5.2] 9.2 dropped, 9.3 deprecated, 9.4+ recommended
  • [5.3] 9.2 dropped, 9.3 deprecated, 9.4+ recommended
  • Rationale: Current releases available for all supported systems from upstream and for CentOS/RHEL 6.x officially via SCL (software collections).
• Python
  • [5.2] 2.6 deprecated and not recommended, 2.7 recommended, 3.x not supported
  • [5.3] 2.6 dropped, 2.7 recommended, 3.x not supported
  • Rationale: 2.7 is provided by all systems except for CentOS/RHEL 6.x, however it is available officially via SCL. Note that 3.3 is also available via SCL, so there is potential for a 3.x migration in the 5.3 timeframe. Dropping 2.6 for 5.3 unifies python support with a single major version, and removes a number of portability issues, but most importantly is a requirement of Django 1.8\(^5\). Python 2.6 will continue to work for 5.2 with Django 1.6, but this is not recommended due to security vulnerabilities.
• GCC
  • [5.2] 4.4 dropped, 4.6+ deprecated, 4.8+ recommended, 5 supported
  • [5.3] 4.6+ dropped, 4.8 and 4.9 supported, 5 recommended
  • Rationale: 4.4 is the CentOS/RHEL 6.x default, 4.8 the 7.x default. The SCL DeveloperToolset3 brings CentOS 6 up to GCC 4.9.1, which makes the new baseline GCC 4.6 (Ubuntu 12.04 and Travis). This will allow use of a limited subset of C++11 features across all platforms in 5.2+. The downside is that while programs and libraries built with the SCL DeveloperToolset3 will generally be link-compatible with the system libraries, use of these libraries with end user code will require the DeveloperToolset3 toolchain and libraries.

The planned changes for 5.2 depend upon our support of CentOS/RHEL 6.x and Ubuntu 12.04.

• LLVM/clang

\(^5\)https://docs.djangoproject.com/en/1.8/releases/1.8/
- [5.2] 3.4 and 3.5 supported, 3.6 recommended and 3.7 supported
- [5.3] 3.4 deprecated, 3.5 and 3.6 supported, 3.7 recommended
- Rationale: It is the current toolchain on MacOS/FreeBSD and is working well; 3.3 dropped due to no longer being used by current toolchains

- **Microsoft Visual Studio**
  - [5.2] 2010 dropped, 2012 deprecated, 2013 recommended,
  - [5.3] 2012 deprecated, 2013 recommended, 2015 supported
  - Rationale: The minimum (and maximum) MSVC version is determined by the Windows Ice builds provided by ZeroC, and as such is limited by our Ice version requirements. Raising the limit in 5.2 will permit use of newer language features.

- **Ice**
  - [5.2] 3.4 dropped, 3.5 recommended, 3.6 supported
  - [5.3] 3.4 dropped, 3.5 deprecated, 3.6 recommended
  - Rationale: 3.4 is no longer supported by ZeroC with 3.5 being current. By the release of 5.3, Ice 3.6 will have been out for some time and should be well tested.

- **Java**
  - [5.2] 6 dropped, 7 deprecated, 8 recommended
  - [5.3] 7 deprecated, 8 recommended
  - Rationale: It will be 2 years this February since the last Java 6 security update. Java 7 is supported on all supported operating systems above, and in most cases is the default Java version on those systems (it is now 8 on some). This was mentioned recently, and also by the Fiji developers; 5.1 would be an opportune time to deprecate or preferably drop Java 6. Also note that we no longer actively test with a Java 6 JVM; all internal testing is on Java 7 or 8 (Oracle and OpenJDK), and there are some client issues with OpenJDK6. Also note that Ice 3.6 drops Java 6 support. For 5.2 deprecate 7; this is because security support ends in April 2015.
  - The requirement for 7 in the Windows clients is less urgent, but will encourage migration to a non-deprecated Java version ahead of the removal of support for 6.

- **Apache**
  - [5.2] deprecated in OMERO 5.2.6
  - [5.3] deprecated, likely to be dropped during the 5.3.x line
  - Rationale: 2.2 is the CentOS/RHEL 6.x and Ubuntu 12.04 version, both of which will be dropped in 5.3 if CentOS 6 is dropped, otherwise they will remain deprecated.

- **mod_wsgi**
  - 3.5+ recommended
  - Rationale: Choice of mod_wsgi version support is made by the OS vendors, who may choose to continue with an older version with security patches applied. For more information refer to mod_wsgi installation\(^6\).

- **nginx**
  - [5.2] 1.4 deprecated, 1.6 supported, 1.8 recommended, 1.10 upcoming
  - [5.3] 1.4 dropped, 1.6 supported, 1.8 supported, 1.10 recommended

### 5.4.2 Support levels

The following sections use the terminology in the table below to describe the support status of a given component, as it progresses from being new and not supported, to supported and tested on a routine basis, and to finally being old and no longer supported nor tested.

\(^6\)https://modwsgi.readthedocs.org/en/develop/installation.html
## 5.4.3 Operating system support

The following subsections detail the versions of each operating system which are supported by both its upstream developers (for security and general updates) and by OME for OMERO building and server and client deployment.

### Microsoft Windows

**General overview** (under ‘W’ in the index)

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
<th>Upstream support (mainline)</th>
<th>Homebrew support</th>
<th>OMERO 5.1</th>
<th>OMERO 5.2</th>
<th>OMERO 5.3</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vista</td>
<td>from Jan 2007</td>
<td>to Apr 2012</td>
<td>to Apr 2017</td>
<td>Dropped</td>
<td>Dropped</td>
<td>Dropped</td>
<td>Ref8</td>
</tr>
<tr>
<td>Win 7</td>
<td>from Oct 2009</td>
<td>to Jan 2015</td>
<td>to Jan 2020</td>
<td>Recommended</td>
<td>Supported</td>
<td>Unsupported†</td>
<td>Ref9</td>
</tr>
<tr>
<td>Server 2008 R2</td>
<td>from May 2008</td>
<td>to Jan 2018</td>
<td>to Jan 2023</td>
<td>Recommended</td>
<td>Supported</td>
<td>Unsupported</td>
<td>Ref10</td>
</tr>
<tr>
<td>Win 8</td>
<td>from Oct 2012</td>
<td>to Jan 2018</td>
<td>to Jan 2023</td>
<td>Supported</td>
<td>Recommended</td>
<td>Unsupported</td>
<td>Ref11</td>
</tr>
<tr>
<td>Win Server 2012</td>
<td>from Oct 2012</td>
<td>to Oct 2020</td>
<td>to Apr 2017</td>
<td>Dropped</td>
<td>Supported</td>
<td>Unsupported</td>
<td>Ref12†</td>
</tr>
<tr>
<td>Win 10</td>
<td>from July 2015</td>
<td>to Apr 2012</td>
<td>to Apr 2017</td>
<td>Dropped</td>
<td>Supported</td>
<td>Unsupported</td>
<td>Ref13</td>
</tr>
</tbody>
</table>

† Unsupported for OMERO.server and OMERO.web deployment. Client support will continue or is upcoming (Windows 10) Insufficient resources are available for regular CI deployment and testing of OMERO.server so a decision has been made to drop support in favor of other priorities. See this blog post for full details.

### UNIX (MacOS X)

**General overview**

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
<th>Upstream support</th>
<th>Homebrew support</th>
<th>OMERO 5.1</th>
<th>OMERO 5.2</th>
<th>OMERO 5.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.9</td>
<td>from Jun 2013</td>
<td>Ended</td>
<td>Ended</td>
<td>Recommended</td>
<td>Supported</td>
<td>Dropped</td>
</tr>
<tr>
<td>10.10</td>
<td>from Oct 2014</td>
<td>Supported</td>
<td>Supported</td>
<td>Recommended</td>
<td>Supported</td>
<td>Unsupported†</td>
</tr>
<tr>
<td>10.11</td>
<td>from Sep 2015</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Dropped</td>
</tr>
<tr>
<td>10.12</td>
<td>from Sep 2016</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Recommended</td>
<td>Deprecated</td>
</tr>
</tbody>
</table>

Apple do not formally announce end of life for their releases, but with the three latest releases being supported this puts 10.10 as the minimum version suitable for use. We have regular CI testing of 10.11 and 10.12 builds plus developer testing of building and client and server deployment. 10.9 and 10.10 are marked as dropped and deprecated since they are no longer tested by the CI infrastructure (nodes retired).

---

7 [http://support.microsoft.com/gp/lifeselectindex](http://support.microsoft.com/gp/lifeselectindex)
8 [http://support.microsoft.com/lifecycle/?p1=11737](http://support.microsoft.com/lifecycle/?p1=11737)
9 [http://support.microsoft.com/lifecycle/?p1=14481](http://support.microsoft.com/lifecycle/?p1=14481)
10 [http://support.microsoft.com/lifecycle/?p1=17383](http://support.microsoft.com/lifecycle/?p1=17383)
11 [http://support.microsoft.com/lifecycle/?p1=16799](http://support.microsoft.com/lifecycle/?p1=16799)
12 [http://support.microsoft.com/lifecycle/?p1=16526](http://support.microsoft.com/lifecycle/?p1=16526)
UNIX (FreeBSD)

It only really makes sense to support the base toolchain for major releases and the Ports tree (which is continually updated); these will be covered in the dependencies, below.

Linux (CentOS and RHEL)

General overview for RHEL\(^{16}\) and CentOS\(^{17}\)

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
<th>Upstream support</th>
<th>OMERO 5.1</th>
<th>OMERO 5.2</th>
<th>OMERO 5.3</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>from Mar 2007</td>
<td>to Mar 2017</td>
<td>Dropped</td>
<td>Dropped</td>
<td>Dropped</td>
<td>Ref(^{18})</td>
</tr>
<tr>
<td>6</td>
<td>from Nov 2010</td>
<td>to Nov 2020</td>
<td>Recommended</td>
<td>Supported</td>
<td>Deprecated</td>
<td>Ref(^{19})</td>
</tr>
<tr>
<td>7</td>
<td>from June 2014</td>
<td>to June 2024</td>
<td>Supported</td>
<td>Supported</td>
<td>Recommended</td>
<td>Ref(^{20})</td>
</tr>
</tbody>
</table>

Only RHEL and CentOS 6 are supported at present. Given the long life of enterprise releases, we intend to support only the latest release at any given time or else it ties us into very old dependencies; 6.x is already quite long in the tooth, however is in wide use and so will require supporting for 5.1 at a minimum. 7.x can be the recommended version once it is properly tested by us and has appropriate CI support. There is currently extensive CI support for building and deployment with CentOS 6 but it is not used directly by developers in general.

Linux (Fedora)

General overview\(^{21}\)

Due to the fast pace of Fedora development, it only really makes sense to support the current and perhaps the previous release; these will be covered in the dependencies, below.

Linux (Ubuntu)

General overview\(^{22}\)

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
<th>Upstream support</th>
<th>OMERO 5.1</th>
<th>OMERO 5.2</th>
<th>OMERO 5.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.04 (.4) LTS</td>
<td>from Apr 2012</td>
<td>to Apr 2017</td>
<td>Deprecated</td>
<td>Dropped</td>
<td>Dropped</td>
</tr>
<tr>
<td>14.04 LTS</td>
<td>from Apr 2014</td>
<td>to Apr 2019</td>
<td>Recommended</td>
<td>Supported</td>
<td>Recommended</td>
</tr>
<tr>
<td>14.10</td>
<td>from Oct 2014</td>
<td>to Jul 2015</td>
<td>Dropped</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>15.04</td>
<td>TBA</td>
<td>TBA</td>
<td>Unsupported</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.10</td>
<td>TBA</td>
<td>TBA</td>
<td>Unsupported</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Only the LTS releases are supported due to resource limitations upon CI and testing. Only the last two LTS releases are supported (being a bit more frequent than CentOS/RHEL). There is currently no CI testing for any version, but some developer use of 12.04 LTS, 14.04 LTS and more recent non-LTS releases.

Linux (Debian stable)

General overview\(^{23}\)

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
<th>Upstream support</th>
<th>OMERO 5.1</th>
<th>OMERO 5.2</th>
<th>OMERO 5.3</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.0</td>
<td>from Feb 2011</td>
<td>to Feb 2016</td>
<td>Dropped</td>
<td>Dropped</td>
<td>Dropped</td>
<td>Reference(^{24})</td>
</tr>
<tr>
<td>7.0</td>
<td>from May 2013</td>
<td>TBA</td>
<td>Recommended</td>
<td>Deprecated</td>
<td>Deprecated</td>
<td>Reference(^{25})</td>
</tr>
<tr>
<td>8.0</td>
<td>TBA</td>
<td>TBA</td>
<td>Unsupported</td>
<td>Supported</td>
<td>Supported</td>
<td>Reference(^{26})</td>
</tr>
</tbody>
</table>

Stable releases 7.x and 6.x are supported. There is no CI testing for any version, and some developer use of 7.x.

\(^{16}\)https://access.redhat.com/site/articles/3078
\(^{17}\)http://wiki.centos.org/FAQ/General#head-fe8a0f0b91ee3c7dea812e8694491e1dde5b75e6d
\(^{18}\)http://wiki.centos.org/FAQ/General#head-fe8a0f0b91ee3c7dea812e8694491e1dde5b75e6d
\(^{19}\)http://wiki.centos.org/FAQ/General#head-fe8a0f0b91ee3c7dea812e8694491e1dde5b75e6d
\(^{20}\)http://wiki.centos.org/FAQ/General#head-fe8a0f0b91ee3c7dea812e8694491e1dde5b75e6d
\(^{21}\)https://fedoraproject.org/wiki/Fedora_Release_Life_Cycle
\(^{22}\)https://wiki.ubuntu.com/Releases
\(^{23}\)https://www.debian.org/releases/
\(^{24}\)https://www.debian.org/releases/squeeze/
\(^{25}\)https://www.debian.org/releases/wheezy/
\(^{26}\)https://www.debian.org/releases/jessie/
Linux (Debian testing and unstable)

Due to the fast pace of Debian development, it only really makes sense to support the last one or two stable releases; the testing and unstable requirements will be covered in the dependencies, below.

5.4.4 Hardware support

Microsoft Windows

Supports amd64 (64-bit) and i386 / x86 (32-bit) for all supported versions; most systems are today running x64 with most hardware being x64, but there is still a niche of x86 users, estimated <25% and falling globally. On Windows, crude estimates are that OMERO.server is 66% amd64, OMERO.insight is 40% amd64, Bio-Formats is 70% amd64. However, note that OMERO.server deployment on Windows is rare, and the i386 systems are not likely to be production servers.

Unix (MacOS X)

Currently all supported versions are amd64 (64-bit) only.

Unix (FreeBSD)

Supports amd64 (64-bit) and i386 / x86 (32-bit) for all supported releases. Other platforms are supported, but may require manual building of prerequisites.

Linux (RHEL and CentOS)

Version 6 supports amd64 (64-bit) and i386 / x86 (32-bit). Version 7 supports amd64 (64-bit) only. Crude estimates (Linux+MacOS) are that OMERO.server is 98% amd64, OMERO.insight is 98% amd64, Bio-Formats is 78% amd64.

Linux (other)

Most distributions support i386 / x86 (32-bit) as well, while some support additional architectures such as 32-bit and/or 64-bit arm and powerpc.

Additional notes

The OMERO server is 64-bit-only due to the large memory requirements. On the client side, there is a requirement for 32-bit support for Windows clients and to a lesser extent on Linux, and zero on MacOS X at present. Dropping 32-bit support for OMERO.server and C++ Ice and other builds should be considered, given the vast reduction in the support burden and steadily dropping need for these builds.

5.4.5 Dependencies

The following subsections detail the versions of each dependency needed by OMERO which are supported by both its upstream developers (for security and general updates) and by OME for OMERO building and server and client deployment.

Note: Versions in brackets are in development distributions and may change without notice.
### Package lists

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS 6 / RHEL 6</td>
<td>Reference(^{27})</td>
</tr>
<tr>
<td>CentOS 7 / RHEL 7</td>
<td>Reference(^{28})</td>
</tr>
<tr>
<td>Fedora (general)</td>
<td>Reference(^{29})</td>
</tr>
<tr>
<td>Fedora 19</td>
<td>Reference(^{30})</td>
</tr>
<tr>
<td>Fedora 20</td>
<td>Reference(^{31})</td>
</tr>
<tr>
<td>Fedora 21</td>
<td>Reference(^{32})</td>
</tr>
<tr>
<td>Ubuntu</td>
<td>Reference(^{33})</td>
</tr>
<tr>
<td>Debian</td>
<td>Reference(^{34})</td>
</tr>
<tr>
<td>Homebrew</td>
<td>Reference(^{35})</td>
</tr>
<tr>
<td>FreeBSD Ports</td>
<td>Reference(^{36})</td>
</tr>
</tbody>
</table>

### PostgreSQL

#### General overview\(^{37}\)

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
<th>Upstream support</th>
<th>OMERO 5.1</th>
<th>OMERO 5.2</th>
<th>OMERO 5.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.2</td>
<td>from Sep 2012</td>
<td>to Sep 2017</td>
<td>Deprecated</td>
<td>Deprecate</td>
<td>Deprecated</td>
</tr>
<tr>
<td>9.3</td>
<td>from Sep 2013</td>
<td>to Sep 2018</td>
<td>Recommended</td>
<td>Deprecated</td>
<td>Deprecate</td>
</tr>
<tr>
<td>9.4</td>
<td>from Dec 2014</td>
<td>to Dec 2019</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>9.5</td>
<td>TBA</td>
<td>TBA</td>
<td>Unsupported</td>
<td>Deprecated</td>
<td>Upcoming</td>
</tr>
</tbody>
</table>

#### Distribution support

<table>
<thead>
<tr>
<th>Version</th>
<th>CentOS/RHEL</th>
<th>Fedora</th>
<th>Ubuntu</th>
<th>Debian</th>
<th>Homebrew</th>
<th>FreeBSD Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.2</td>
<td>7.x</td>
<td>19</td>
<td>N/A</td>
<td>N/A</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>9.3</td>
<td>N/A</td>
<td>20, 21</td>
<td>14.04</td>
<td>N/A</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>9.4</td>
<td>N/A</td>
<td>N/A</td>
<td>14.10</td>
<td>(8.0)</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>9.5</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

The PostgreSQL project provides packages\(^{42}\) for supported platforms. Therefore distribution support is not critical since 9.3 and 9.4 are available for all platforms.

---

\(^{27}\)http://mirror.centos.org/centos/6/os/x86_64/Packages/
\(^{28}\)http://mirror.centos.org/centos/7/os/x86_64/Packages/
\(^{29}\)https://fedoraproject.org/wiki/Releases
\(^{30}\)https://admin.fedoraproject.org/pkgdb/packages/*/?branches=f19&status=&owner=
\(^{31}\)https://admin.fedoraproject.org/pkgdb/packages/*/?branches=f20&status=&owner=
\(^{32}\)https://admin.fedoraproject.org/pkgdb/packages/*/?branches=f21&status=&owner=
\(^{33}\)http://packages.ubuntu.com/search?keywords=foo&searchon=names&suite=all&section=all
\(^{34}\)https://packages.debian.org/search?keywords=postgresql&searchon=sourcenames&suite=all&section=all
\(^{35}\)https://github.com/Homebrew/homebrew-core/tree/master/Formula
\(^{36}\)http://svnweb.freebsd.org/ports/head/
\(^{37}\)http://www.postgresql.org/support/versioning/
\(^{38}\)http://www.postgresql.org/support/versioning/
\(^{39}\)https://apps.fedoraproject.org/packages/postgresql
\(^{40}\)http://packages.ubuntu.com/search?keywords=postgresql&searchon=names&suite=all&section=all
\(^{41}\)https://packages.debian.org/search?keywords=postgresql&searchon=sourcenames&suite=all&section=all
\(^{42}\)http://www.postgresql.org/download/
OMERO Documentation, Release 5.2.6

Python

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
<th>Upstream support</th>
<th>OMERO 5.1</th>
<th>OMERO 5.2</th>
<th>OMERO 5.3</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.6</td>
<td>from Oct 2008</td>
<td>to Oct 2013</td>
<td>Deprecated</td>
<td>Deprecated</td>
<td>Dropped</td>
<td>Reference[43]</td>
</tr>
<tr>
<td>2.7</td>
<td>from Jul 2010</td>
<td>to 2020</td>
<td>Recommended</td>
<td>Recommended</td>
<td>Recommended</td>
<td>Reference[44]</td>
</tr>
<tr>
<td>3.2</td>
<td>from Feb 2011</td>
<td>to Feb 2016</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Reference[45]</td>
</tr>
<tr>
<td>3.3</td>
<td>from Sep 2012</td>
<td>to Sep 2017</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Reference[46]</td>
</tr>
<tr>
<td>3.4</td>
<td>from Mar 2014</td>
<td>TBA</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Reference[47]</td>
</tr>
<tr>
<td>3.5</td>
<td>TBA</td>
<td>TBA</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Reference[48]</td>
</tr>
</tbody>
</table>

Distribution support

<table>
<thead>
<tr>
<th>Version</th>
<th>CentOS/RHEL</th>
<th>Fedora</th>
<th>Ubuntu</th>
<th>Debian</th>
<th>Homebrew</th>
<th>FreeBSD Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.6</td>
<td>6.x</td>
<td>N/A</td>
<td>10.04</td>
<td>6.0, 7.0</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>2.7</td>
<td>7.x</td>
<td>19, 20, 21</td>
<td>12.04, 13.04, 13.10, 14.04, 14.10, (15.04)</td>
<td>7.0, (8.0)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3.2</td>
<td>N/A</td>
<td>N/A</td>
<td>12.04</td>
<td>7.0</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>3.3</td>
<td>N/A</td>
<td>19, 20</td>
<td>13.04, 13.10</td>
<td>N/A</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>3.4</td>
<td>N/A</td>
<td>21</td>
<td>14.04, 14.10, (15.04)</td>
<td>(8.0)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3.5</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

At the moment 2.7 support is present upstream for the foreseeable future; 3.x versions continue to be released and retired regularly in parallel. The limiting factor will be distribution support for 2.7 as major packages are slowly switching to 3.x, and this might cause problems if our python module dependencies are no longer available without major effort. Ice 3.5 in particular has dropped 2.7 support for Windows, and has a significant cost in providing custom rebuilds; this has been rectified in Ice 3.6 with the Python module being installable with pip using any Python version.

The supported version of the Django module used by OMERO.web (1.8) requires Python 2.7. The older version (1.6) will work with Python 2.6 but lacks security support, and is consequently not recommended for production use.

GCC

General overview[55]

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
<th>Upstream support</th>
<th>OMERO 5.1</th>
<th>OMERO 5.2</th>
<th>OMERO 5.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4</td>
<td>from Apr 2009</td>
<td>to Mar 2012</td>
<td>Deprecated</td>
<td>Deprecated</td>
<td>Dropped</td>
</tr>
<tr>
<td>4.5</td>
<td>from Apr 2010</td>
<td>to Jul 2012</td>
<td>Deprecated</td>
<td>Deprecated</td>
<td>Dropped</td>
</tr>
<tr>
<td>4.6</td>
<td>from Mar 2011</td>
<td>to Apr 2013</td>
<td>Supported</td>
<td>Supported</td>
<td>Dropped</td>
</tr>
<tr>
<td>4.7</td>
<td>from Mar 2012</td>
<td>to Apr 2013</td>
<td>Supported</td>
<td>Supported</td>
<td>Dropped</td>
</tr>
<tr>
<td>4.8</td>
<td>from Mar 2013</td>
<td>to May 2014</td>
<td>Recommended</td>
<td>Recommended</td>
<td>Dropped</td>
</tr>
<tr>
<td>4.9</td>
<td>from Apr 2014</td>
<td>to Jun 2015</td>
<td>Recommended</td>
<td>Recommended</td>
<td>Dropped</td>
</tr>
<tr>
<td>5.0</td>
<td>from Apr 2015</td>
<td>TBA</td>
<td>Unsupported</td>
<td>Supported</td>
<td>Recommended</td>
</tr>
</tbody>
</table>

5.4. Version requirements
## Distribution support

<table>
<thead>
<tr>
<th>Version</th>
<th>CentOS/RHEL</th>
<th>Fedora</th>
<th>Ubuntu</th>
<th>Debian</th>
<th>Homebrew</th>
<th>FreeBSD Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4</td>
<td>6.x</td>
<td>N/A</td>
<td>10.04, 12.04, 13.04, 13.10, 14.04, 14.10</td>
<td>6.0, 7.0</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>4.5</td>
<td>N/A</td>
<td>N/A</td>
<td>12.04</td>
<td>N/A</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>4.6</td>
<td>N/A</td>
<td>N/A</td>
<td>12.04, 13.04, 13.10, 14.04, 14.10</td>
<td>7.0, (8.0)</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>4.7</td>
<td>N/A</td>
<td>N/A</td>
<td>13.04, 13.10, 14.04, 14.10</td>
<td>7.0, (8.0)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>4.8</td>
<td>7.x</td>
<td>19, 20</td>
<td>13.10, 14.04, 14.10, (15.04)</td>
<td>(8.0)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>4.9</td>
<td>N/A</td>
<td>21</td>
<td>14.04, 14.10, (15.04)</td>
<td>(8.0)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>5.0</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

GCC 4.2 support was dropped with the dropping of MacOS 10.6; the current baseline is GCC 4.4.

## LLVM/clang

### General overview

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
<th>Upstream support</th>
<th>OMERo 5.1</th>
<th>OMERo 5.2</th>
<th>OMERo 5.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4</td>
<td>from Jan 2014 to May 2014</td>
<td>from Jan 2014 to May 2014</td>
<td>Recommended</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>3.5</td>
<td>from Sep 2014</td>
<td>to Sep 2014</td>
<td>Supported</td>
<td>Recommended</td>
<td>Supported</td>
</tr>
<tr>
<td>3.6</td>
<td>from Feb 2015</td>
<td>to Jul 2015</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Supported</td>
</tr>
<tr>
<td>3.7</td>
<td>from Sep 2015</td>
<td>TBA</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

### Distribution support

<table>
<thead>
<tr>
<th>Version</th>
<th>CentOS/RHEL</th>
<th>Fedora</th>
<th>Ubuntu</th>
<th>Debian</th>
<th>Homebrew</th>
<th>FreeBSD Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4</td>
<td>N/A</td>
<td>20, 21</td>
<td>12.04, 13.10, 14.04, 14.10, (15.04)</td>
<td>(8.0)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3.5</td>
<td>N/A</td>
<td>(22)</td>
<td>14.10, (15.04)</td>
<td>(8.0)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3.6</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>(8.0)</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>3.7</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Note that clang++ 3.4 is used in FreeBSD 10.1 and MacOS 10.9 and 10.10. clang++ 3.6 is used in MacOS 10.10 and 10.11 (Xcode 7). Also note that the MacOS version deviates from the official releases and has some features disabled or crippled such as the address sanitizer and OpenMP.

## Microsoft Visual Studio

### General overview

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
<th>Upstream (mainline)</th>
<th>OMERo 5.1</th>
<th>OMERo 5.2</th>
<th>OMERo 5.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>from Jun 2010</td>
<td>to Jul 2015</td>
<td>Deprecated</td>
<td>Supported</td>
<td>Deprecated</td>
</tr>
<tr>
<td>2012</td>
<td>from Oct 2012</td>
<td>to Jan 2018</td>
<td>Deprecation</td>
<td>Deprecation</td>
<td>Supported</td>
</tr>
<tr>
<td>2013</td>
<td>from Jan 2014</td>
<td>to Apr 2019</td>
<td>Deprecated</td>
<td>Recommended</td>
<td>Supported</td>
</tr>
<tr>
<td>2015</td>
<td>from Jun 2015</td>
<td>to Oct 2020</td>
<td>Deprecated</td>
<td>Unsupported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

56https://apps.fedoraproject.org/packages/gcc
57http://packages.fedoraproject.org/packages/g++
58http://packages.debian.org/search?keywords=g%2B%2B&searchon=names&suite=all&section=all
59http://llvm.org/releases/
60https://apps.fedoraproject.org/packages/clang
61http://packages.ubuntu.com/search?keywords=clang&searchon=names&suite=all&section=all
62https://packages.debian.org/search?keywords=clang&searchon=names&suite=all&section=all
63http://support2.microsoft.com/lifecycle/search/?sort=PN&alphas=Visual+Studio

5.4. Version requirements 72
The version to use is largely dependent upon the Ice support for a particular Visual Studio version.

- Ice 3.6 supports 2015, 2013 and 2012
- Ice 3.5 supports 2013, 2012 and 2010
- Ice 3.4 supports 2010 and 2008

In general later versions have better C++ standards conformance and correctness, and should be preferred where possible.

### Ice

#### General overview

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
<th>Upstream support</th>
<th>OMERO 5.1</th>
<th>OMERO 5.2</th>
<th>OMERO 5.3</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3</td>
<td>from May 2008 to May 2009</td>
<td>Dropped</td>
<td></td>
<td></td>
<td></td>
<td>Ice 3.35 Ice 3.3.166</td>
</tr>
<tr>
<td>3.4</td>
<td>from Mar 2010 to Jun 2011</td>
<td>Deprecated</td>
<td>Dropped</td>
<td></td>
<td></td>
<td>Ice 3.467 Ice 3.4.268</td>
</tr>
<tr>
<td>3.5</td>
<td>from Mar 2013 to Oct 2013</td>
<td>Recommended</td>
<td>Dropped</td>
<td></td>
<td></td>
<td>Ice 3.569 Ice 3.5.170</td>
</tr>
<tr>
<td>3.6</td>
<td>from June 2015 to TBA</td>
<td>Unsupported</td>
<td>Dropped</td>
<td></td>
<td></td>
<td>Ice 3.6.071 Ice 3.6.172 Ice 3.6.2</td>
</tr>
</tbody>
</table>

#### Distribution support

<table>
<thead>
<tr>
<th>Version</th>
<th>CentOS/RHEL</th>
<th>Fedora</th>
<th>Ubuntu</th>
<th>Debian</th>
<th>Homebrew</th>
<th>FreeBSD Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3</td>
<td>N/A</td>
<td>N/A</td>
<td>10.04</td>
<td>6.0</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>3.4</td>
<td>N/A</td>
<td>N/A</td>
<td>12.04, 13.04, 13.10</td>
<td>7.0</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>3.5</td>
<td>N/A</td>
<td>19, 20, 21</td>
<td>14.04, 14.10, (15.04)</td>
<td>(8.0)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3.6</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### Java

#### General overview

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
<th>Upstream support</th>
<th>OMERO 5.1</th>
<th>OMERO 5.2</th>
<th>OMERO 5.3</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>from Dec 2006 to Feb 2013</td>
<td>Deprecated</td>
<td>Dropped</td>
<td></td>
<td></td>
<td>Reference64</td>
</tr>
<tr>
<td>7</td>
<td>From Jul 2011 to Apr 2015</td>
<td>Supported</td>
<td>Deprecated</td>
<td></td>
<td></td>
<td>Reference65</td>
</tr>
<tr>
<td>8</td>
<td>From Mar 2014 to Mar 2017</td>
<td>Recommended</td>
<td>Recommended</td>
<td></td>
<td></td>
<td>Reference66</td>
</tr>
</tbody>
</table>

64https://zeroc.com/download.html
65https://forums.zeroc.com/forum/announcements/3851-ice-3-3-released
66https://forums.zeroc.com/forum/announcements/4362-ice-3-3-1-released
67https://forums.zeroc.com/forum/announcements/4946-ice-3-4-released
68https://forums.zeroc.com/forum/announcements/5541-ice-3-4-2-released
69https://forums.zeroc.com/forum/announcements/6093-ice-3-5-0-released
70https://forums.zeroc.com/forum/announcements/6283-ice-3-5-1-released
71https://forums.zeroc.com/forum/announcements/45941-ice-3-6-0-and-ice-touch-3-6-0-released
72https://forums.zeroc.com/forum/announcements/46347-ice-ice-e-and-ice-touch-3-6-1-released
73https://forums.zeroc.com/forum/announcements/46347-ice-ice-e-and-ice-touch-3-6-1-released
74http://www.oracle.com/technetwork/java/eol-135779.html
75http://www.oracle.com/technetwork/java/eol-135779.html
76http://www.oracle.com/technetwork/java/eol-135779.html
77http://www.oracle.com/technetwork/java/eol-135779.html
Distribution support

<table>
<thead>
<tr>
<th>Version</th>
<th>CentOS/RHEL</th>
<th>Fedora</th>
<th>Ubuntu</th>
<th>Debian</th>
<th>Homebrew</th>
<th>FreeBSD Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>6.x, 7.x</td>
<td>N/A</td>
<td>10.04, 12.04, 13.04, 13.10, 14.04, 14.10, (15.04)</td>
<td>6.0, 7.0, (8.0)</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>6.x, 7.x</td>
<td>19, 20, 21, (22)</td>
<td>12.04, 13.04, 13.10, 14.04, 14.10, (15.04)</td>
<td>7.0, (8.0)</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>N/A</td>
<td>19, 20, 21, (22)</td>
<td>14.10, (15.04)</td>
<td>(8.0)</td>
<td>N/A</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Note that all distributions provide OpenJDK 6, 7 and/or 8 due to distribution restrictions by Oracle. Oracle Java may be used if downloaded separately.

Oracle no longer allow general downloading of Java 6. Java 7 is available for all platforms except for OSX 10.6 (where it is installable manually by unpacking the 10.7 installer). Some 1.8 OpenJDK versions are broken (Debian/Ubuntu 1.8.0u40 broken), but some work (1.8.0u25, FreeBSD). 1.8 should become recommended once working versions exist in all distributions.

OpenGL

General overview

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
<th>Upstream support</th>
<th>OMERO 5.1</th>
<th>OMERO 5.2</th>
<th>OMERO 5.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>in Sep 2004</td>
<td>N/A</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Unsupported</td>
</tr>
<tr>
<td>2.1</td>
<td>in Jun 2006</td>
<td>N/A</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Unsupported</td>
</tr>
<tr>
<td>3.0</td>
<td>in Aug 2008</td>
<td>N/A</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Unsupported</td>
</tr>
<tr>
<td>3.1</td>
<td>in Mar 2009</td>
<td>N/A</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Unsupported</td>
</tr>
<tr>
<td>3.2</td>
<td>in Aug 2009</td>
<td>N/A</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Unsupported</td>
</tr>
<tr>
<td>3.3</td>
<td>in Mar 2010</td>
<td>N/A</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Unsupported</td>
</tr>
<tr>
<td>3.4</td>
<td>in Jun 2010</td>
<td>N/A</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Unsupported</td>
</tr>
<tr>
<td>4.0</td>
<td>in Mar 2010</td>
<td>N/A</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Unsupported</td>
</tr>
<tr>
<td>4.1</td>
<td>in Aug 2010</td>
<td>N/A</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Unsupported</td>
</tr>
<tr>
<td>4.2</td>
<td>in Aug 2010</td>
<td>N/A</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Unsupported</td>
</tr>
<tr>
<td>4.3</td>
<td>in Aug 2012</td>
<td>N/A</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Unsupported</td>
</tr>
<tr>
<td>4.4</td>
<td>in Jul 2013</td>
<td>N/A</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Unsupported</td>
</tr>
<tr>
<td>4.5</td>
<td>in Aug 2014</td>
<td>N/A</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td>Unsupported</td>
</tr>
</tbody>
</table>

78 https://apps.fedoraproject.org/packages/java-1.7.0-openjdk
79 https://admin.fedoraproject.org/pkgdb/package/java-1.8.0-openjdk/
80 http://packages.ubuntu.com/search?keywords=jdk&searchon=names&suite=all&section=all
81 https://packages.debian.org/search?keywords=jdk&searchon=names&suite=all&section=all
82 http://en.wikipedia.org/wiki/OpenGL

5.4. Version requirements
### Distribution support

<table>
<thead>
<tr>
<th>Version</th>
<th>Windows</th>
<th>CentOS/RHEL</th>
<th>Fedora</th>
<th>Ubuntu</th>
<th>Debian</th>
<th>MacOS</th>
<th>FreeBSD Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>Varies*</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>N/A</td>
<td>Varieties†</td>
</tr>
<tr>
<td>2.1</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>10.7, 10.8, 10.9, 10.10</td>
<td>Varieties†</td>
</tr>
<tr>
<td>3.0</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>N/A</td>
<td>Varieties†</td>
</tr>
<tr>
<td>3.1</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>V/A</td>
<td>Varieties†</td>
</tr>
<tr>
<td>3.2</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>10.7, 10.8 (core profile)</td>
<td>Varieties†</td>
</tr>
<tr>
<td>3.3</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>10.9, 10.10 (core profile min)</td>
<td>Varieties†</td>
</tr>
<tr>
<td>4.0</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>N/A</td>
<td>Varieties†</td>
</tr>
<tr>
<td>4.1</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>10.9, 10.10 (core profile max)</td>
<td>Varieties†</td>
</tr>
<tr>
<td>4.2</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>N/A</td>
<td>Varieties†</td>
</tr>
<tr>
<td>4.3</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>N/A</td>
<td>Varieties†</td>
</tr>
<tr>
<td>4.4</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>N/A</td>
<td>Varieties†</td>
</tr>
<tr>
<td>4.5</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>Varies†</td>
<td>N/A</td>
<td>Varieties†</td>
</tr>
</tbody>
</table>

Details

* Driver/hardware dependent, but the Windows Vista/7/8 DirectX 9 requirement is mostly the same as OpenGL 2.0, so any badged machine should have OpenGL 2.0-capable hardware (it will also require a functional driver)

† Driver/hardware dependent

Support is largely down to the system hardware (GPU) and drivers. There are no guarantees for 2.0 and greater on Windows (1.4 only guaranteed), but AMD, Intel and nVidia all provide GL drivers supporting newer versions. The minimum baseline is probably 2.1. However, 3.2/3.3 is probably the baseline for most hardware under 5 years old, with 4.x being the typical baseline for hardware from a year ago.

### Apache

#### General overview and website

Version | Release date | Upstream support | OMERO 5.1 | OMERO 5.2 | OMERO 5.3 |
--------|--------------|------------------|-----------|-----------|-----------|
2.2     | from Dec 2005| TBA              | Supported | Deprecated| Deprecated|
2.4     | from Feb 2012| TBA              | Recommended| Deprecated| Deprecated|

Apache 2.2+ requires mod_wsgi which is included in the standard distribution on CentOS 7 (or has to be installed from external repository, for CentOS6). For more information refer to mod_wsgi installation.

### nginx

#### General overview and roadmap

Details

5.4. Version requirements
OMERO Documentation, Release 5.2.6

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
<th>Upstream support</th>
<th>OMERO 5.1</th>
<th>OMERO 5.2</th>
<th>OMERO 5.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.4</td>
<td>from Apr 2013</td>
<td>to Mar 2014</td>
<td>Supported</td>
<td>Deprecated</td>
<td>Dropped</td>
</tr>
<tr>
<td>1.6</td>
<td>from Apr 2014</td>
<td>TBA</td>
<td>Recommended</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>1.8</td>
<td>from Apr 2015</td>
<td>TBA</td>
<td>Unsupported</td>
<td>Recommended</td>
<td>Supported</td>
</tr>
<tr>
<td>1.10</td>
<td>from April 2016</td>
<td>TBA</td>
<td>Unsupported</td>
<td>Upcoming</td>
<td>Recommended</td>
</tr>
</tbody>
</table>

**Distribution support**

<table>
<thead>
<tr>
<th>Version</th>
<th>CentOS/RHEL</th>
<th>Fedora</th>
<th>Ubuntu</th>
<th>Debian</th>
<th>Homebrew</th>
<th>FreeBSD Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.4</td>
<td>N/A</td>
<td>N/A</td>
<td>19, 20</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>1.6</td>
<td>EPEL</td>
<td>21, (22)</td>
<td>13.10, 14.04</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>1.8</td>
<td>N/A</td>
<td>N/A</td>
<td>14.10, (15.04)</td>
<td>N/A</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>1.10</td>
<td>N/A</td>
<td>N/A</td>
<td>16.04</td>
<td>N/A</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>Details</td>
<td></td>
<td>N/A</td>
<td>Ref92</td>
<td>Ref93</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 5.5 Example production server set-ups

#### 5.5.1 Micron, Oxford

The OMERO server at Micron, Oxford94 houses two OMERO instances, the databases for both these instances, and a single OMERO.web instance which serves them both. The second OMERO instance (Raff OMERO) originated from another group’s private OMERO server, which is now managed by Micron, but there was no way to merge this data into the main server. The main OMERO instance is configured to interface to a departmental LDAP server to authenticate users and get initial authorization details.

OMERO Data1 in the diagram is a large filestore server which hosts all the image data. This is made available to the OMERO server itself via a Samba mount. This server has 36 TiB of space of which OMERO is using 16 TiB and Raff OMERO is using 600 GiB. This is backed up to a tape robot.

OMERO Processor1 consists of a 32 core, 128GiB RAM processing machine for doing image analysis. This is connected on a completely private network to the OMERO server (to avoid issues with configuring OMERO.grid to be secure) and runs scripts using OMERO.grid.

**Stats**

- 90 users
- 40 groups
- 36 TiB of data storage space, of which 16.6 TiB is currently in use
- Performance statistics to come

#### 5.5.2 Biozentrum, University of Basel

The OMERO server at Biozentrum95 has around 82 users and currently uses 9.8 TB of data storage space, with an average monthly increase of 200-300 GB. It is run on CentOS 6.5 with data hosted on an NFS-mounted NAS system.

---

92http://packages.ubuntu.com/search?keywords=nginx&searchon=names&suite=all&section=all
93https://packages.debian.org/search?keywords=nginx&searchon=names&suite=all&section=all
94http://www2.bioch.ox.ac.uk/microngroup/micron_home.php
95http://www.biozentrum.unibas.ch
Hardware

Remote storage consists of:

- NFS-mounted NAS system

Local storage consists of:

- 2 x 146 GB 6Gbit/s SAS HD, RAID 1, OS and OMERO software
- 2 x 128 GB SSD, RAID 1, Postgres DB
- 2 x 600 GB 6Gbit/s SAS HD, scratch space, buffer, etc.

Computational resources:

- IBM x3550 M4, dual-CPU Intel(R) Xeon(R) E5-2643 3.3GHz, physical
- 32 GB of memory

Network infrastructure

- 1 Gbit/s Ethernet, local, VPN support, dedicated
5.5.3 GReD Research Center, Clermont-Ferrand, France

The Genetics, Reproduction and Development Research Center\footnote{https://www.gred-clermont.fr} has 65 users and currently uses 3 TB of storage, with an average monthly increase of 90 GB. It is run on Debian Squeeze.

**Hardware**

- 11 TB of storage spread over 8 local hard drives (2 TB), RAID 5

**Computational resources:**

- 1 Intel Xeon E5506 (4 physical cores)
- 8 GB of memory

**Network infrastructure**

The server is hosted inside the faculty of medicine where the network works at 100 Mbit/s. There are are 4 Gbit/s ports on the server but only one is currently in use.

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

5.6 Known limitations

5.6.1 Time zone

We do not recommend changing the time zone on your server. The server is currently set to use local time and changing time zones will result in a mismatch between the original data import times stored in the server and the way the clients report them. A fix for this is forthcoming but will not be in 5.1.

5.6.2 Java issues

OpenJDK8 is supported, however some builds of some releases (such as the current Ubuntu build of 8u40) have broken JPEG support, reported for OpenJDK\footnote{http://icedtea.classpath.org/bugzilla/show_bug.cgi?id=1393} and Ubuntu\footnote{https://bugs.launchpad.net/ubuntu/+source/openjdk-8/+bug/1421501}. See \url{#12612} and \url{this one-user's thread}\footnote{https://trac.openmicroscopy.org/ome/ticket/12612} for more information. OracleJDK8 is also supported, but the earliest releases have had problems. The latest OpenJDK8 or OracleJDK8 should work correctly (we have checked both OpenJDK 1.8.0_45-internal and OracleJDK 1.8.0_66). OpenJDK and OracleJDK version 7 are also supported (we have tested OracleJDK 1.7.0_80 and OpenJDK 1.7.0_85).

5.6.3 Windows OS issues

**Failure response on import for new databases**

If you have import failures with an error such as:

```
java.lang.RuntimeException: Failure response on import!
Category: ::omero::grid::ImportRequest
Name: import-request-failure
Parameters: {stacktrace=java.lang.RuntimeException: omero.ValidationException
serverStackTrace = "ome.conditions.ValidationException: could not
```

\footnote{https://lists.openmicroscopy.org.uk/pipermail/ome-users/2015-November/005723.html}
When trying to import data to a new database (or possibly one just upgraded to 5.1), it may be an issue with the language encoding during database initialization leading to corrupted Unicode characters, such that any import containing these characters (e.g. µ) will fail with the above exception. A `SET CLIENT_ENCODING TO 'UTF8';` command has been added to the database initialization instructions on OMERO.server installation to prevent this.

**UTF-8 character limitations in passwords**

Windows does not handle non-Latin characters well in the command prompt (console). If you use Unicode characters in your passwords, you may find they are not correctly formed. Therefore, we recommend you avoid them if at all possible.

**Binary delete**

On Windows servers not all binary files corresponding to a delete may be removed from the binary repository. See [Binary data](#) for more details.

**5.6.4 Ubuntu issues**

**Importing using desktop clients**

Under older Ubuntu installations, the ‘import folder’ option in the desktop clients currently does not work.

**5.6.5 Windows issues**

**C++ compilation problems**

OMERO.cpp will not currently build on Windows due to exceeding DLL symbol limits on this platform, leading to a failure when linking the DLL. It is hoped that this platform limitation can be worked around in a future OMERO release.

**5.6.6 Searching**

**5.6.7 Too many open file descriptors**

Starting with OMERO 5, the server works directly from original files. At times, this requires a significant number of open file handles. If you are having problems with large or frequent imports, are seeing “Too many open file descriptors” or similar, you may need to increase the maximum number of open files per process. On Linux, this may be done by setting the `nofile` limit in `/etc/security/limits.conf`, for example:

```bash
omero soft nofile 10000
omero hard nofile 12000
```

This permits the `omero` user to have 10000 open files per process, which may be increased up to a maximum of 12000 by the user. The username and limits will need adjusting for the specifics of your installation and usage requirements. Note that these settings take effect only for new logins, so the server and the shell or environment the server is started from will require restarting. Run `ulimit -a` as the user running OMERO to verify that the changes have taken effect.
5.6.8 Deleting datasets

If you have a dataset containing any images which are also present in any other datasets, you will not be able to delete it. First, you must remove the shared images by cutting the links.

5.6.9 Changing group permissions

If a group contains a projection made by one member from data owned by another user, you cannot make the group into a private group.

5.6.10 File format support

Large images with floating-point pixel data

Pyramids of image tiles are currently not generated for images with floating-point pixel data, meaning the imported image will be scrambled if it is over a certain size (configurable using omero.pixeldata.max_plane_height and omero.pixeldata.max_plane_width but set to 3192x3192 pixels by default). This primarily affects the following file formats:

- Gatan DM3
- MRC
- TIFF

Calculation of minima and maxima pixel values

If images are imported with one of the omero import --skip options skipping calculation of the global minima and maxima pixel values, OMERo clients will use the extrema of the pixel type range by default. Users can adjust the minima/maxima via the rendering settings. Recalculating minima and maxima pixel values after import is currently not supported.

Flex data in OMERo.tables

If you are using the advanced configuration setting FlexReaderServerMaps for importing Flex data split between multiple directories for use with OMERo.tables, you should not upgrade beyond 5.0.x. Neither the 5.1 line nor OMERo 5.2 support this functionality.

5.6.11 LDAP

Enabling synchronization of LDAP on user login may override admin actions carried out in the clients, see Synchronizing LDAP on user login for details.

Note: This documentation is for OMERo 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERo 5.3 is expected before the end of 2016.

5.7 Groups and permissions system

See also:
OMERO permissions history, querying and usage
5.7.1 Summary

A user may belong to one or more groups, and the data in a group may now at most be shared with users in the same group on the same OMERO server. The degree to which their data is available to other members of the group depends on the permissions settings for that group. Whenever a user logs on to an OMERO server, they are connected under one of their groups. All data they import and any work that is done is assigned to the current group, however the user can now easily copy their data into another group.

5.7.2 Users

Administrator  Your OMERO server will have one or more administrators. Each group can be administrated by any of your server administrators. The administrators control all settings for groups.

Group owner  Your group may have one or more owners. The group owner has some additional rights within each group than a standard group member, including the ability to add other members to the group.

Group member  This is the standard user.

Groups and users must be created by the server administrator. Users can then be added by the administrator or by one of the group owners assigned by the administrator. This would typically be the PI of the lab. The group’s owners or server administrator can also choose the permission level for that group. See the Help guide for managing groups for more information about how to administrate them in OMERO.

5.7.3 Group permission levels

The various permission levels are:

Private  This group is the most restrictive:

- A private Group owner can see and control who the group members are and can view their data.
- As a Group member, you will only ever be able to see your own data.
- This can be used for general data storage, access and analysis, but has very limited collaboration potential other than for the Group owner to see other group members’ data.

Potential use cases of Private group:

- A PI as Group owner and their student, as a Group member, can access the student’s data. A student might use this as somewhere to store all of their data and from here, the PI and/or student might decide which data could/should be copied into a more collaborative group where additional members would also be able to view the data.
- An institutional repository type structure where data are being archived, but not necessarily open for general viewing.

Read-only  This group is the intermediate option that allows visibility of other users and their data, but minimal ability to annotate their data:

- The Group owner can control group members as above and can perform some annotations on the other group members data.
- Group member can see who other members are and view their data, but cannot annotate another members’ data at all.

Potential use cases of Read-only group:

- A scientist might move data into a read-only group when they want other group members to access and view their data. Their PI, as a group owner could then annotate and/or add Regions of Interest (ROIs) to their images.
- For an institutional repository where data are being archived and then available for other users in the institute to view; this could be standard storage of all original data, or for data that is included in publications.

Read-annotate  This is a more collaborative group:

- Group member can view other members, their data and can make annotations on those other members’ data.

Potential use cases of Read-annotate group:

- This could be used by a group of scientists working together with data for a publication.

101 http://help.openmicroscopy.org/sharing-data.html#owner
Read-write  This group essentially allows all the group members to behave as if they co-own all the data:

- Group member can view, annotate, edit and delete all data; the only restriction is that they cannot move other members’ data into another group.

Potential use cases of Read-write group:

- A group of scientists working in a completely collaborative way, trusting every member of the group to have equal rights and access to all the data.

See also:

Help guide for sharing data[^102]  Workflow guide covering the groups and permissions system

### 5.7.4 Changing group permissions

It is possible for the Group owner or server Administrator to change the permissions level on a group after it has been created and filled with data, with the following limitations:

- It is not possible to ‘reduce’ permissions to Private if the group contains a projection made by one member from data owned by another user. In other circumstances, reducing permissions to private will warn of loss of annotations etc. as noted below, but will still be possible.

- Only Administrator can promote a group to Read-write permissions. Make certain all the members understand that this allows anyone in the group to permanently delete any of the data before performing this action.

**Warning:** Please be very careful before downgrading a group’s permission level. If a user has annotated other users’ data and the group is downgraded, any links to annotations that are not permitted by the new permissions level will be lost.

### 5.7.5 Permissions on your and other users’ data

**What can you do with your data?**

All OMERO users in all groups can perform all actions on their own data.

The main actions available include, but are not limited to:

- create projects and/or datasets
- import data
- delete data
- edit names and descriptions of images
- change rendering settings on images
- annotate images (rate, tag, add attachments and comments)
- de-annotate (remove annotations that you have added)
- use Regions of Interest (ROIs) (add, import, edit, delete, save and analyze them)
- run scripts
- move data between groups, if you belong to more than one group

**What can you do with someone else’s data in your group?**

Actions available for you on someone else in your group’s data will depend both on the permissions of the group you are working in, and what sort of user you are. See the table below for a quick reference guide to permissions available on other people’s data.

Some of these policies may evolve as the permissions functionality matures in response to user feedback. Please let us know any comments or suggestions you have via our [mailing lists][103] or through the [forums][104].

[^102]: http://www.openmicroscopy.org/site/community/mailing-lists
[^103]: http://www.openmicroscopy.org/site/community/mailing-lists
[^104]: http://www.openmicroscopy.org/community/
5.7.6 Permissions tables

The following are the permissions valid for users working on data belonging to other group members. These permissions depend on the group permissions and on the type of the user performing the action.

### Administrator

<table>
<thead>
<tr>
<th>Action</th>
<th>Private</th>
<th>Read-only</th>
<th>Read-annotate</th>
<th>Read-write</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Annotate</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Delete</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Edit</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Move between groups</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Remove annotations</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Mix data</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

### Group owner

<table>
<thead>
<tr>
<th>Action</th>
<th>Private</th>
<th>Read-only</th>
<th>Read-annotate</th>
<th>Read-write</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Annotate</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Delete</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Edit</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Move between groups</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Remove annotations</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Mix data</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

### Group member

<table>
<thead>
<tr>
<th>Action</th>
<th>Private</th>
<th>Read-only</th>
<th>Read-annotate</th>
<th>Read-write</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Annotate</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Delete</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Edit</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Move between groups</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Remove annotations</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Mix data</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>
Key

Action  Action on other users’ data

Annotate  Add annotations (rating, tag, attachment, comment ROI) to another users’ data. Also create & save ROIs (save ROIs that you draw on another users’ data).

Delete  Delete data such as images or ROIs. ROIs may have been added by others or yourself.

Edit  Modify the name or description of other users’ objects such as images.

Mix data  Copy, Move or Remove other users’ data to or from your Projects, Datasets or Screens. Copy, Move or Remove your or others’ data to or from others’ Projects, Datasets or Screens.

Note:  You should always be able to remove annotations (such as tags) that you linked to other users’ data (you own the link). The link can be deleted, but the tag itself will not be deleted.

Move between groups  Only the admin has the right to move other users’ data between groups.

Note:  The admin does not have to be member of the destination group.

Remove annotations  Remove annotations made by others on your data.

Render  Create your own rendering settings (this will not modify the settings of the owner).

View  View other users’ data such as images. View ROIs added by others. Draw ROIs on other users’ data, but they cannot be saved.

5.7.7 Issues to be aware of

ROIs

• You can never edit (change text or move) other users’ ROI.

• Any ROIs added to other users’ data will not affect ROIs added by the owner.

Tags and attachments

• A tag or attachment is ‘owned’ by the person who creates it or uploads it to the server.

• The link between a tag or an attachment is ‘owned’ by the person who annotates an image with that tag or attachment i.e. makes a link between the tag/attachment and the image.

• De-annotation deletes the link between the tag/attachment and image but does not remove/delete the tag or attachment from the system.

Scripts

• Although all users can run scripts on other users’ data, the actions within those scripts will be subject to the restrictions of the permissions detailed in the tables above.
This chapter contains instructions for installing OMERO.server, and for troubleshooting, backing-up, and upgrading your server installation.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.1 Server installation on UNIX (and UNIX-like platforms)

The pages below provide guidance on how to install and set up OMERO.server on any of the supported UNIX and UNIX-like platforms. Specific walkthroughs are provided for several systems, with detailed step-by-step instructions. However, reading through the main OMERO.server installation guide first is recommended as this gives an overview of the entire process and introduces the additional pages listed below.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.1.1 OMERO.server installation

This section covers the installation of OMERO.server on UNIX and UNIX-like platforms. This includes all BSD, Linux and Mac OS X systems. Depending upon which platform you are using, you may find a more specific walk-through listed below.

See also:

OMERO.server installation on CentOS 6 with Python 2.7 and Ice 3.5  Instructions for installing OMERO.server from scratch on CentOS 6 with Python 2.7 and Ice 3.5.

OMERO.server installation on CentOS 6 with Python 2.7 and Ice 3.6  Instructions for installing OMERO.server from scratch on CentOS 6 with Python 2.7 and Ice 3.6.

OMERO.server installation on CentOS 7 and Ice 3.5  Instructions for installing OMERO.server from scratch on CentOS 7 with Ice 3.5.

OMERO.server installation on CentOS 7 and Ice 3.6  Instructions for installing OMERO.server from scratch on CentOS 7 with Ice 3.6.

OMERO.server installation on Ubuntu 14.04 and Ice 3.5  Instructions for installing OMERO.server from scratch on Ubuntu 14.04 with Ice 3.5.

OMERO.server installation on Ubuntu 14.04 and Ice 3.6  Instructions for installing OMERO.server from scratch on Ubuntu 14.04 with Ice 3.6.

OMERO.server installation on OS X with Homebrew  Instructions for installing and building OMERO.server on Mac OS X with Homebrew using our special formulas (i.e. from the source code via Homebrew). You do not need to refer to this page to install the prerequisites with Homebrew and then install the server zip from the downloads page.
Prerequisites

Installation will require:

- a clean, minimal operating system installation
- a “root” level account for which you know the password

Note: If you are unsure of what it means to have a “root” level account, or if you are generally having issues with the various users/passwords described in this install guide, please see Which user account and password do I use where?.

The installation and configuration of the prerequisite applications are mostly outside the scope of this document. For Linux distributions, use of the default package manager is recommended. For BSD systems, the ports system provides all the prerequisites. For MacOS X, Homebrew is recommended. This guide provides the package names to install for a number of contemporary systems. However, the names and versions provided vary between releases. Please do check for similar packages if the one documented here is not available for your system as it may be provided under an alternative name. “Debian” refers to Debian and derivative distributions such as Ubuntu. “RedHat” refers to RedHat and related distributions such as CentOS, Fedora and Scientific Linux.

- For Ubuntu you need to enable the universe repository. This should be enabled by default. If not enabled, it may be enabled by editing /etc/apt/sources.list directly, in which case the entries may already exist but are commented out, or by using Synaptic (10.04 and 10.10) or Ubuntu Software Center (11.04 onwards). Update your package lists to ensure that you get the latest packages:

  $ sudo apt-get update

  Install packages by running:

  $ sudo apt-get install package

  where package is the package name to install.

The following subsections cover the details for each package, in the order recommended for installation.

Java SE Runtime Environment (JRE)

If possible, install one of the following packages:

<table>
<thead>
<tr>
<th>System</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSD Ports</td>
<td>java/openjdk7</td>
</tr>
<tr>
<td>Debian</td>
<td>openjdk-7-jre</td>
</tr>
<tr>
<td>Homebrew</td>
<td>N/A (install Oracle Java)</td>
</tr>
<tr>
<td>RedHat</td>
<td>java-1.8.0-openjdk</td>
</tr>
</tbody>
</table>

OMERO works with the OpenJDK JRE provided by most systems, or with Oracle Java. Version 1.7 or later is required.

Your system may already provide a suitable JRE, in which case no extra steps are necessary. Linux distributions usually provide OpenJDK, and older MacOS X versions have Java installed by default. Oracle Java is no longer provided by BSD or Linux distributions for licensing reasons. If your system does not have Java available, for example on newer MacOS X versions, or the provided version is too old, Oracle Java may be downloaded from the Oracle website

Warning: Security

Installing Oracle Java outside the system’s package manager will leave your system without regular distribution-supplied security updates, and so is not recommended.

To check which version of Java is currently available:

$ which java
/usr/bin/java
$ java --version
java version "1.7.0_51"
Java(TM) SE Runtime Environment (build 1.7.0_51-b13)
Java HotSpot(TM) 64-Bit Server VM (build 24.51-b03, mixed mode)

Python 2

Check you have Python (and check its version) by running:

$ python --version
Python 2.7.10

Note: OMERO does not currently support Python 3; you must use 2.7.

The following Python packages are required:

<table>
<thead>
<tr>
<th>Package</th>
<th>Functionality</th>
<th>Downloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Django(^1) (1.8)(^3)</td>
<td>OMERO web</td>
<td>Django(^4) page</td>
</tr>
<tr>
<td>Pillow(^5)(^6)</td>
<td>OMERO web and Figure Export</td>
<td>Pillow(^7) page</td>
</tr>
<tr>
<td>Matplotlib(^8)</td>
<td>OMERO web</td>
<td>Matplotlib page(^9)</td>
</tr>
<tr>
<td>NumPy (1.2.0 or higher)(^10)</td>
<td>Scripting</td>
<td>Numpy/Scipy page(^11)</td>
</tr>
<tr>
<td>PyTables (2.1.0 or higher)(^12)</td>
<td>OMERO.tables</td>
<td>PyTables page(^12)</td>
</tr>
</tbody>
</table>

Note: Some of these can be ignored if you wish to forego some functionality but we recommend you just install everything. For example, scripting is enabled by default so should not be assumed optional even if you never expect your users to run scripts from the clients.

If possible, install the following packages:

<table>
<thead>
<tr>
<th>System</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSD Ports</td>
<td>lang/python27 graphics/py-pillow math/py-matplotlib math/py-numpy devel/py-tables science/py-scipy</td>
</tr>
<tr>
<td>Debian</td>
<td>python2.7 python-pil python-matplotlib python-numpy python-tables python-scipy</td>
</tr>
<tr>
<td>Homebrew</td>
<td>python pillow numpypillowmatplotlib</td>
</tr>
<tr>
<td>RedHat</td>
<td>python2.7 python-pil python-matplotlib python-numpy python-tables python-scipy</td>
</tr>
</tbody>
</table>

Note: CentOS 6 users should read OMERO.server installation on CentOS 6 with Python 2.7 and Ice 3.5 or OMERO.server installation on CentOS 6 with Python 2.7 and Ice 3.6 and follow the instructions there to install Python and the required modules.

Ice

Note: OMERO 5.2 supports 3.5 and 3.6 on UNIX and UNIX-like platforms. You must install the correct version of OMERO.server (see Downloads\(^13\)).

3https://www.djangoproject.com/
3The currently supported version of the django module used by OMERO.web (1.8) requires Python 2.7. The older version (1.6) will work with Python 2.6 but lacks security support, and is consequently not recommended for production use. Python 2.7 and Django 1.8 (https://docs.djangoproject.com/en/1.8/releases/1.8/) are required for security support.
4https://pypi.python.org/pypi/Django
5http://pillow.readthedocs.org
6Make sure to have libjpeg (http://libjpeg.sourceforge.net/) installed when building Pillow (http://pillow.readthedocs.org). We currently do not support version 3.0+
7https://pypi.python.org/pypi/Pillow
8http://matplotlib.org/
9http://matplotlib.org/
10May already have been installed as a dependency of Matplotlib.
11http://www.scipy.org/Download
13http://downloads.openmicroscopy.org/latest/omero/
The Ice version may vary, depending upon the distribution version you are using. The Ice versions in currently supported versions
of Debian and Ubuntu are shown in the Ice of the Version requirements page.

Using the latest version of Ice is recommended, where possible. If your package manager provides Ice packages, using these
is recommended where possible. Distribution-provided packages often have additional bugfixes which are not present in the
upstream releases.

If needed, source and binary packages are available from ZeroC. The latest release is available from the ZeroC website.

Note: CentOS 6 users should read OMERO.server installation on CentOS 6 with Python 2.7 and Ice 3.5 or OMERO.server
installation on CentOS 6 with Python 2.7 and Ice 3.6 and follow the instructions there to install Ice.

Note: ZeroC Ice can always be built from source code for specific platforms if a binary package is not available.

Note: With Ice 3.6, the Python bindings are provided separately. If your package manager does not provide Ice python packages,
run pip install zeroc-ice to install the Ice Python bindings. See Using the Python Distribution for further details.

OMERO.web

Please install one of apache (and its wsgi module) or nginx and Gunicorn in order to run OMERO.web.

<table>
<thead>
<tr>
<th>System</th>
<th>Packages</th>
</tr>
</thead>
</table>
| BSD Ports| www/apache24 www/nginx
| Debian   | apache2 libapache2-mod-wsgi nginx
| Homebrew | nginx              |
| RedHat   | httpd mod_wsgi nginx |

OMERO.scripts

If you wish to run the “Movie Maker” script, please install mencoder.

<table>
<thead>
<tr>
<th>System</th>
<th>Packages</th>
</tr>
</thead>
</table>
| BSD Ports| multimedia/mencoder
| Debian   | mencoder         |
| Homebrew | mplayer          |
| RedHat   | mencoder         |

Installation

Once the above prerequisites have been downloaded, installed and configured appropriately, the OMERO server itself may be
installed. You may wish to create a user account solely for the purpose of running the server, and switch to this user for the next
steps.

Server directory

Firstly, a directory needs to be created to contain the server. In this case ~/omero is used as an example:

$ mkdir -p ~/omero

Next, change into this directory:
$ cd ~/omero

OMERO.server

The release OMERO.server.zip is available from the OMERO downloads\(^19\) page. Download the version matching the version of Ice installed on your system before continuing.

Installing a development version from source is also possible. See the Installing OMERO from source section for further details. This is not recommended unless you have a specific reason not to use a release version.

Once you have obtained the OMERO.server zip archive matching the version of Ice installed, unpack it:

$ unzip OMERO.server-5.2.6-ice35-byy.zip

If your system does not provide an `unzip` command by default, install one of the following:

<table>
<thead>
<tr>
<th>System</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSD Ports</td>
<td>archivers/unzip</td>
</tr>
<tr>
<td>Debian</td>
<td>unzip</td>
</tr>
<tr>
<td>Homebrew</td>
<td>unzip</td>
</tr>
<tr>
<td>RedHat</td>
<td>unzip</td>
</tr>
</tbody>
</table>

Optionally, give your OMERO software install a short name to save some typing later, to reflect what you set `OMERO_PREFIX` to in the Environment variables section, below:

$ ln -s OMERO.server-5.2.6-ice35-byy OMERO.server

This will also ease installation of newer versions of the server at a later date, by simply updating the link.

Environment variables

If using distribution-provided packages such as Debian or RPM packages, or via the homebrew or macports package manager, it should not be necessary to set any environment variables. However, if using third-party packages for any required components, several variables may require setting in order for them to function correctly.

Please note that the precise details of these environment variables can change as new versions of software are released.

There are several methods for setting environment variables; which is most appropriate will depend upon how the OMERO server is started. Options include:

\[\text{/etc/security/pam_env.conf} \text{ Global environment set at login by PAM}\]

\[\text{/etc/profile or /etc/profile.d/omero} \text{ Global Bourne shell defaults (also used by derived shells such as bash and zsh)}\]

\[\text{~/.profile} \text{ User's Bourne shell defaults (also used by derived shells)}\]

\[\text{/etc/bash.bashrc} \text{ Global bash defaults}\]

\[\text{~/.bashrc, ~/.bash_profile or ~/.bash_login} \text{ User's bash configuration.}\]

If OMERO is started as a service using an init script, a global setting should be preferred. If being started by hand using a particular user, a user-specific configuration file may be more appropriate.

The following environment variables may be configured:

\[\text{LD_LIBRARY_PATH (Linux) or DYLD_LIBRARY_PATH (MacOS X)} \text{ The Ice and PostgreSQL libraries must be on the library search path. If using the packages provided by your distribution, this will already be the case. If using third-party binary distributions the lib (or lib64 if present and using a 64-bit system) directory for each will require adding to the library search path.}\]

\[\text{OMERO_PREFIX} \text{ This is not strictly required, but may be set for convenience to point to the OMERO server installation, and is used in this documentation as a shorthand for the installation path.}\]

\[\text{OMERO_TMPDIR} \text{ Directory used for temporary files. If the home directory of the user running the OMERO server is located on a slow filesystem, such as NFS, this may be used to store the temporary files on fast local storage.}\]

\(^{19}\)http://downloads.openmicroscopy.org/latest/omero/

6.1. Server installation on UNIX (and UNIX-like platforms) 89
 PATH The search path must include the programs java, python, icegridnode and PostgreSQL commands such as psql. If using the packages provided by your distribution, this will already be the case. If using third-party binary distributions such as the ZeroC Ice package, Oracle Java, or PostgreSQL, the bin directory for each must be added to the path. The OMERO bin directory may also be added to the search path ($OMERO_PREFIX/bin if OMERO_PREFIX has been set).

 PYTHONPATH The Ice python directory must be made available to python. If using the Ice packages provided by your distribution, this will already be the case. If using the ZeroC ice package, add the python directory to the python path. For Ice 3.6, this should never be required.

 Warning: The OMERO_HOME environment variable is used internally by OMERO. Unless you really know what you are doing, it is strongly recommended that you do not set this variable (see Setting the OMERO_HOME environment variable for details). You can use a different name of your choice instead, indicated by OMERO_PREFIX in this documentation.

 After making any needed changes, either source the corresponding file or log back in for them to take effect. Run env to check them.

 Note: CentOS 6 users should read OMERO.server installation on CentOS 6 with Python 2.7 and Ice 3.5 or OMERO.server installation on CentOS 6 with Python 2.7 and Ice 3.6 and set the needed environment variables as documented.

 Creating a database

 On most systems, a “postgres” user will be created which has admin privileges, while the UNIX root user itself does not have admin privileges. Therefore it is necessary to either become the postgres user, or use sudo as shown below.

 For the purposes of this guide, the following dummy data is used:

 Username: db_user
 Password: db_password
 Database: omero_database

 Warning: Security
 These dummy values are examples only and should not be used. For a live or public server install these values should be altered to reflect your security requirements—i.e. use your own choice of username and password instead. These should not be the same username and/or password as your Linux/Mac/Windows root user!
 You should also consider restricting access to your server machine, but that is outside the scope of this document.

 • Create a non-superuser database user and record the name and password used. You will need to configure OMERO to use this username and password later on:

   $ sudo -u postgres createuser -P -D -R -S db_user
   Enter password for new role: # db_password
   Enter it again: # db_password

 • Create a database for OMERO to reside in:

   $ sudo -u postgres createdb -E UTF8 -O db_user omero_database

 • Check to make sure the database has been created, you have PostgreSQL client authentication correctly set up and the database is owned by the db_user user.

   $ psql -h localhost -U db_user -l
   Password for user db_user:
   List of databases
   Name                  | Owner | Encoding
   ----------------------|-------|-------------
   omero_database        | db_user | UTF8
   postgres               | postgres | UTF8

6.1. Server installation on UNIX (and UNIX-like platforms)
If you have problems, especially with the last step, take a look at OMERO.server and PostgreSQL since the authentication mechanism is probably not properly configured.

**Location for the your OMERO binary repository**

- Create a directory for the OMERO binary data repository. /OMERO is the default location and should be used unless you explicitly have a reason not to and know what you are doing.
- This is *not* where you want the OMERO application to be installed, it is a separate directory which the OMERO.server will use to store binary data.
- You can read more about the OMERO binary repository.

```
$ sudo mkdir /OMERO
```

- Change the ownership of the directory. /OMERO must either be owned by the user starting the server (it is currently owned by the system root) or that user must have permission to write to the directory. You can find out your username and edit the correct permissions as follows:

```
$ whoami
omer
$ sudo chown -R omero /OMERO
```

**Configuration**

- Optionally, review ~/.omero/OMERO.server/etc/omero.properties, which contains all default settings. Do not edit this file—it is for reference only. Any configuration settings you would like to change can be changed in the next step. Alternatively, you can view a parsed version of the file under Configuration properties glossary or parse it yourself with omero config parse.
- Change any settings that are necessary using omero config, including the name and/or password for the ‘db_user’ database user you chose above or the database name if it is not “omero_database”. (Quotes are only necessary if the value could be misinterpreted by the shell. See link\(^{20}\))

```
$ cd ~/.omero/OMERO.server
$ bin/omero config set omero.db.name 'omero_database'
$ bin/omero config set omero.db.user 'db_user'
$ bin/omero config set omero.db.pass 'db_password'
```

You can also check the values that have been set using:

```
$ cd ~/.omero/OMERO.server
$ bin/omero config get
```

- If you have chosen a non-standard OMERO binary repository location above, be sure to configure the omero.data.dir property. For example, to use /srv/omero:

```
$ omero config set omero.data.dir /srv/omero
```

• Create the OMERO database initialization script. You will need to provide a password for the newly created OMERO root user, either by using the --password argument or by entering it when prompted. Note that this password is for the root user of the OMERO.server, and is not related to the root system user or a PostgreSQL user role.

```
$ cd ~/omero/OMERO.server
$ bin/omero db script --password omero_root_password
```

Using OMERO5.2 for version
Using 0 for patch
Using password from commandline
Saving to /home/omero/OMERO5.2__0.sql

**Warning: Security**
For illustrative purposes, the default password for the OMERO root user is shown as `omero_root_password`. However, you should **not** use this default values for your installation, but use your own choice of password instead. This should **not** be the same password as your Linux/Mac/Windows root user or the database user!

• Initialize your database with the script.

```
$ psql -h localhost -U db_user omero_database < OMERO5.2__0.sql
```
At this point you should see some output from PostgreSQL as it installs the schema for new OMERO database.

• Before starting the OMERO.server, run the OMERO diagnostics script to check that all of the settings are correct, e.g.

```
$ bin/omero admin diagnostics
```

• You can now start the server using:

```
$ bin/omero admin start
Creating var/master
Initializing var/log
Creating var/registry
No descriptor given. Using etc/grid/default.xml
```

• If multiple users have access to the system running OMERO you should restrict access to the OMERO.server/etc and OMERO.server/var directories, for example by changing the permissions on them:

```
$ chmod 700 ~/omero/OMERO.server/etc ~/omero/OMERO.server/var
```

You should also consider restricting access to the OMERO data repository. The required permissions will depend on whether you are using **Advanced import scenarios**.

• Test that you can log in as “root”, either with the OMERO.insight client or on the command-line:

```
$ bin/omero login
Server: [localhost]
Username: [root]
Password: # omero_root_password
```

You will be prompted for an OMERO username and password. Use the username and password set when running bin/omero db script.

• If your users are going to be importing many files in one go, for example multiple plates, you should make sure you set the maximum number of open files to a sensible level (i.e. at least 8K for production systems, 16K for bigger machines). See **Too many open files** for more information.

6.1. Server installation on UNIX (and UNIX-like platforms) 92
**JVM memory settings**

The OMERO server starts a number of Java services. Memory settings for these are calculated on a system-by-system basis. An attempt has been made to have usable settings out of the box, but if you can afford to provide OMERO with more memory, it will certainly improve your overall performance. See *Memory configuration* on how to tune the JVM.

**OMERO.web and administration**

In order to deploy OMERO.web in a production environment such as Apache or Nginx please follow the instructions under *OMERO.web deployment*.

**Note:** The internal Django webserver can be used for evaluation and development. In this case please follow the instructions under *OMERO.web deployment for developers*.

**Enabling movie creation from OMERO.**

OMERO has a facility to create AVI/MPEG Movies from images. The page *OMERO.movie* details how to enable it.

**Post-installation items**

**Backup**

One of your first steps after putting your OMERO server into production should be deciding on when and how you are going to backup your database and binary data. Please do not omit this step.

**Security**

It is also now recommended that you read the *Server security and firewalls* page to get a good idea as to what you need to do to get OMERO clients speaking to your newly installed OMERO.server in accordance with your institution or company’s security policy.

**Advanced configuration**

Once you have the base server running, you may want to try enabling some of the advanced features such as *OMERO.dropbox* or *LDAP authentication*. If you have **Flex data**, you may want to watch the HCS configuration screencast\(^{21}\). See the Feature list\(^{22}\) for more advanced features you may want to use, and *Configuration properties glossary* on how to get the most out of your server.

**Troubleshooting**

My OMERO install doesn’t work! What do I do now? Examine the *Troubleshooting OMERO* page and if all else fails post a message to our ome-users\(^{23}\) mailing list discussed on the *Community support* page. Especially the *Server fails to start* and *Remote clients cannot connect to OMERO installation* sections are a good starting point.

**OMERO diagnostics**

If you want help with your server installation, please include the output of the diagnostics command:

```
$ bin/omero admin diagnostics
```

---


\(^{22}\) [http://www.openmicroscopy.org/site/products/omero/feature-list](http://www.openmicroscopy.org/site/products/omero/feature-list)

OMERO Documentation, Release 5.2.6

Commands: java -version 1.7.0 (/usr/bin/java)
Commands: python -V 2.7.9 (/usr/bin/python)
Commands: icegridnode --version 3.5.1 (/usr/bin/icegridnode)
Commands: icegridadmin --version 3.5.1 (/usr/bin/icegridadmin)
Commands: psql --version 9.4.5 (/usr/bin/psql)

Server: icegridnode running
Server: Blitz-0 active (pid = 30324, enabled)
Server: DropBox active (pid = 30343, enabled)
Server: FileServer active (pid = 30345, enabled)
Server: Indexer-0 active (pid = 30348, enabled)
Server: MonitorServer active (pid = 30351, enabled)
Server: OMERO.Glacier2 active (pid = 30353, enabled)
Server: OMERO.IceStorm active (pid = 30376, enabled)
Server: Processor-0 active (pid = 30394, enabled)
Server: Tables-0 inactive (disabled)
Server: TestDropBox inactive (enabled)

OMERO: SSL port 4064
OMERO: TCP port 4063

Log dir: /home/omero/OMERO.server-5.0.1-ice35-b21/var/log exists

Log files:
Blitz-0.log 3.0 MB errors=0 warnings=9
DropBox.log 4.0 KB errors=0 warnings=1
FileServer.log 0.0 KB
Indexer-0.log 10.0 KB errors=0 warnings=5
MonitorServer.log 2.0 KB
OMERO.web.log 642.0 KB errors=0 warnings=1
OMERO.web_request.log 0.0 KB
PixelData-0.log 7.0 KB errors=0 warnings=4
Processor-0.log 2.0 KB errors=0 warnings=1
Tables-0.log n/a
TestDropBox.log n/a
master.err 0.0 KB errors=2 warnings=0
master.out 0.0 KB
Total size 3.83 MB

Environment:
OMERO_HOME=(unset)
OMERO_NODE=(unset)
OMERO_MASTER=(unset)
OMERO_TEMPDIR=(unset)
PATH=/usr/local/bin:/usr/bin:/bin
ICE_HOME=(unset)
LD_LIBRARY_PATH=(unset)

OMERO SSL port:4064
OMERO TCP port:4063
OMERO data dir: '/OMERO' Exists? True Is writable? True
OMERO temp dir: '/home/omero/tmp' Exists? True Is writable? True (Size: 0)

JVM settings:
Blitz-0(index) -Xmx621m -XX:MaxPermSize=512m
Indexer-0(index) -Xmx414m -XX:MaxPermSize=512m
PixelData-0(index) -Xmx621m -XX:MaxPermSize=512m

6.1. Server installation on UNIX (and UNIX-like platforms) 94
JVM settings: Repository-$(index) -Xmx414m -XX:MaxPermSize=512m -XX:+IgnoreUnrecognizedVMOptions

OMERO.web status... You are deploying OMERO.web using apache and mod_wsgi. Cannot check status.
Django version: 1.8.7

Update notification

Your OMERO.server installation will check for updates each time it is started from the Open Microscopy Environment update server. If you wish to disable this functionality you should do so now as outlined on the OMERO upgrade checks page.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.1.2 OMERO.server installation on CentOS 6 with Python 2.7 and Ice 3.5

This installation walkthrough should be read in conjunction with OMERO.server installation and OMERO.web deployment.

Running OMERO on CentOS 6 has a number of special requirements which deviate from the standard installation instructions. The instructions below will set up Python 2.7 and Ice 3.5 on CentOS 6. We tested the installation with Python 2.7 from IUS and used a virtual environment to install the various dependencies required to install an OMERO.server. It is also possible to use SCL Python (for example walkthrough_centos6_py27.sh) but such solution could have potential side effects.

Setting up

Python 2.7

CentOS 6 provides Python 2.6. However, OMERO.web requires Python 2.7 in order to use Django 1.8. While Django 1.6 may be used with Python 2.6, this version of Django no longer has security support. In consequence, it is necessary to upgrade to Python 2.7 in order to obtain Django security updates, which are required for a production deployment.

Ice 3.5

The RPM packages provided by ZeroC for CentOS 6 use the Python 2.6 provided by the system. OMERO will need an Ice build which uses Python 2.7. A version of Ice 3.5 built against Python 2.7 installed above is available at http://downloads.openmicroscopy.org/ice/experimental. Ice will be installed into /opt.

Installing prerequisites

The following steps are run as root.

Install Java 1.8, Ice 3.5 and PostgreSQL 9.4:

To install Java 1.8 and other dependencies:

```
# epel-release will be pulled as a dependency
yum -y install https://centos6.iuscommunity.org/ius-release.rpm

# installed for convenience
yum -y install unzip wget tar

# install Java
```
yum -y install java-1.8.0-openjdk

# install dependencies
yum -y install \python27 \python27-devel \libjpeg-devel \libpng-devel \libtiff-devel \hdf5-devel \zlib-devel \freetype-devel

# install pip and virtualenv using Python 2.6
yum -y install python-pip

pip install --upgrade virtualenv

# if virtualenv is not installed (unlikely)
# yum -y install python27-pip
# pip2.7 install virtualenv

# TODO: this installs a lot of unnecessary packages:
yum -y groupinstall "Development Tools"

export PYTHONWARNINGS="ignore:Unverified HTTPS request"

To install Ice 3.5:

curl -o /etc/yum.repos.d/zeroc-ice-el6.repo \http://download.zeroc.com/Ice/3.5/el6/zeroc-ice-el6.repo

yum -y install db53 db53-utils mcpp
mkdir /tmp/ice-download
cd /tmp/ice-download

wget http://downloads.openmicroscopy.org/ice/experimental/Ice-3.5.1-b1-centos6-iuspy27-x86_64.tar.gz
tar -zxvf /tmp/ice-download/Ice-3.5.1-b1-centos6-iuspy27-x86_64.tar.gz

# Install under /opt
mv Ice-3.5.1-b1-centos6-iuspy27-x86_64 /opt/Ice-3.5.1

# make path to Ice globally accessible
# if globally set, there is no need to export LD_LIBRARY_PATH
echo /opt/Ice-3.5.1/lib64 > /etc/ld.so.conf.d/ice-x86_64.conf
ldconfig

To install PostgreSQL 9.4:

# install Postgres
# Postgres, reconfigure to allow TCP connections
yum -y install http://yum.postgresql.org/9.4/redhat/rhel-6-x86_64/pgdg-centos94-9.4-2.noarch.rpm
yum -y install postgresql94-server postgresql94

service postgresql-9.4 initdb
sed -i.bak -re 's/^host.*ident/\1md5/ ' /var/lib/pgsql/9.4/data/pg_hba.conf
chkconfig postgresql-9.4 on
service postgresql-9.4 start
The remaining dependencies will be installed in a virtual environment:

```
# Install the OMERO dependencies in a virtual environment
# Create virtual env.
# -p only required if virtualenv has been installed with python 2.6
virtualenv -p /usr/bin/python2.7 /home/omero/omeroenv
set +u
source /home/omero/omeroenv/bin/activate
set -u
/home/omero/omeroenv/bin/pip install --upgrade pip
/home/omero/omeroenv/bin/pip2.7 install -r requirements_centos6_py27_ius.txt
deactivate
```

See `requirements_centos6_py27_ius.txt`

Create an omero system user, and a directory for the OMERO repository:

```
useradd -m omero
mkdir -p "$OMERO_DATA_DIR"
chown omero "$OMERO_DATA_DIR"
```

Create a database user and initialize a new database for OMERO:

```
echo "CREATE USER $OMERO_DB_USER PASSWORD '$OMERO_DB_PASS'"  | \
   su - postgres -c psql
su - postgres -c "createdb -E UTF8 -O '$OMERO_DB_USER' '$OMERO_DB_NAME'"
psql -P pager=off -h localhost -U "$OMERO_DB_USER" -l
```

The following settings will need adding to your OMERO startup script or to the omero user’s environment (for example in a shell startup script). Add the absolute path to the `bin` directory of the virtual environment `/home/omero/omeroenv` to the `PATH` variable:

```
# Environment file for OMERO
export ICE_HOME=/opt/Ice-3.5.1
export PATH="$ICE_HOME/bin:/home/omero/omeroenv/bin:$PATH"
#Remove commented out export below if Ice is not set globally accessible
#export LD_LIBRARY_PATH="$ICE_HOME/lib64:$ICE_HOME/lib:$LD_LIBRARY_PATH"
export PYTHONPATH="$ICE_HOME/python:$PYTHONPATH"
export SLICEPATH="$ICE_HOME/slice"
```

These settings will enable Python 2.7, and set the necessary environment variables for Ice 3.5 to work.

For example, download `omero-centos6py27ius.env` into `/home/omero` and run:

```
echo source ~/.ome/omero-centos6py27ius.env >> ~/.ome/.bashrc
```

Install OMERO.server

The following steps are run as the omero system user.

Download, unzip and configure OMERO. The rest of this walkthrough assumes the OMERO.server is installed into the home directory of the omero system user.
Note that this script requires the same environment variables that were set earlier in settings.env, so you may need to copy and/or source this file as the omero user.

You will need to install the server corresponding to your Ice version.

Install server-ice35.zip:

```
cd ~omero
SERVER=http://downloads.openmicroscopy.org/latest/omero5.2/server-ice35.zip
wget $SERVER
unzip -q OMERO.server*
```

Configure:

```
ln -s OMERO.server-*/* OMERO.server
OMERO.server/bin/omero config set omero.data.dir "$OMERO_DATA_DIR"
OMERO.server/bin/omero config set omero.db.name "$OMERO_DB_NAME"
OMERO.server/bin/omero config set omero.db.user "$OMERO_DB_USER"
OMERO.server/bin/omero config set omero.db.pass "$OMERO_DB_PASS"
OMERO.server/bin/omero db script -f OMERO.server/db.sql --password "$OMERO_ROOT_PASS"
psql -h localhost -U "$OMERO_DB_USER" "$OMERO_DB_NAME" < OMERO.server/db.sql
```

**Installing a web server**

To Deploy OMERO.web, you can use either Nginx or Apache. Follow the steps to install your chosen web server.

**Nginx**

See also [OMERO.web Nginx and Gunicorn deployment (Unix/Linux)](OMERO.web Nginx and Gunicorn deployment (Unix/Linux)).

**The following steps are run as root.**

Install Nginx 1.8, install the requirements to run OMERO.web in the virtual environment, deactivate it and copy the Nginx OMERO configuration file into the Nginx configuration directory, and disable the default configuration:

```
cat << EOF > /etc/yum.repos.d/nginx.repo
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=0
enabled=1
EOF

#install nginx
yum -y install nginx

virtualenv -p /usr/bin/python2.7 /home/omero/omeroenv
set +u
source /home/omero/omeroenv/bin/activate
set -u

# Install OMERO.web requirements
/home/omero/omeroenv/bin/pip2.7 install -r -omoermo/OMERO.server/share/web/requirements-py27-nginx.txt
deactivate

# set up as the omero user.
su - omero -c "bash -euo setup_omero_nginx.sh"

cmp /etc/nginx/conf.d/default.conf /etc/nginx/conf.d/default.disabled
```
OMERO Documentation, Release 5.2.6

```
cp ~omero/OMERO.server/nginx.conf.tmp /etc/nginx/conf.d/omero-web.conf

service nginx start

Apache

See also DEPRECATED: OMERO.web Apache and mod_wsgi deployment (Unix/Linux).

As the omero system user, configure OMERO.web by running the following commands:

OMERO.server/bin/omero config set omero.web.application_server wsgi
OMERO.server/bin/omero web config apache --http "$OMERO_WEB_PORT" > OMERO.server/apache.conf.tmp
OMERO.server/bin/omero web syncmedia

The following steps are run as root.

The version of Apache installed is 2.2, see the note below if you wish to use version 2.4.

Install Apache 2.2, install the requirements to run OMERO.web in the virtual environment, deactivate it and add the path to the virtual environment python to the python-path parameter of the WSGIDaemonProcess directive:

```
#install Apache 2.2
yum -y install httpd
# install mod_wsgi compiled against 2.7
yum -y install python27-mod_wsgi

virtualenv -p /usr/bin/python2.7 /home/omero/omeroenv
set +u
source /home/omero/omeroenv/bin/activate
set -u

# Install OMERO.web requirements
/home/omero/omeroenv/bin/pip2.7 install -r ~omero/OMERO.server/share/web/requirements-py27-apache.txt
deactivate

# Add virtual env python to the python-path parameter of the WSGIDaemonProcess directive
sed -i 's/\([^/\s]*\)/\1\//g' /etc/httpd/conf.d/omero-web.conf

chkconfig httpd on

service httpd start
```

**Note:** If you wish to use Apache 2.4 with mod_wsgi, see [omero-install](https://github.com/ome/omero-install) for a possible solution.

Running OMERO

The following steps are run as the omero system user.

OMERO should now be set up. To start the server run:

```
OMERO.server/bin/omero admin start
```

[27](https://github.com/ome/omero-install)
If you deploy with Nginx, to start the OMERO.web client run:

```
OMERO.server/bin/omero web start
```

The last command is **not** necessary if you deploy with Apache. Nginx or Apache should already be running so you should be able to log in as the OMERO root user by going to `http://localhost/` in your web browser. Please read the **SELinux** section below.

In addition some example *init.d* scripts are available should you wish to start OMERO and OMERO.web automatically:

```
cp omero-init.d /etc/init.d/omero
chmod a+x /etc/init.d/omero

cp omero-web-init.d /etc/init.d/omero-web
chmod a+x /etc/init.d/omero-web

chkconfig --del omero
chkconfig --add omero
chkconfig --del omero-web
chkconfig --add omero-web
```

```
omero-init.d
omero-web-init.d
```

### Securing OMERO

**The following steps are run as root.**

If multiple users have access to the machine running OMERO you should restrict access to OMERO.server’s configuration and runtime directories, and optionally the OMERO data directory:

```
chmod go-rwx ~omero/OMERO.server/etc ~omero/OMERO.server/var
```

```
# Optionally restrict access to the OMERO data directory
# chmod go-rwx "$OMERO_DATA_DIR"
```

### Regular tasks

**The following steps are run as root.**

The default OMERO.web session handler uses temporary files to store sessions which should be deleted at regular intervals, for instance by creating a cron job:

```
OMERO_USER=omero
OMERO_SERVER=/home/omero/OMERO.server
su - ${OMERO_USER} -c "${OMERO_SERVER}/bin/omero web clearsessions"
```

Copy the following commands into the appropriate location:

```
cp omero-web-cron /etc/cron.daily/omero-web
chmod a+x /etc/cron.daily/omero-web
```

```
omero-web-cron
```
SELinux

The following steps are run as root.

If you are running a system with SELinux enabled (it is enabled by default on CentOS 6)\(^{28}\) and are unable to access OMERO.web you may need to adjust the security policy:

```bash
if [ $(getenforce) != Disabled ]; then
  yum -y install policycoreutils-python
  setsebool -P httpd_read_user_content 1
  setsebool -P httpd_enable_homedirs 1
  semanage port -a -t http_port_t -p tcp 4080
fi
```

Installing Web apps

The following steps are run as root.

It is possible to add Web applications to OMERO. If your app required some extra Python packages installed using `pip`, those packages should be also installed in the virtual environment. For example, OMERO.figure\(^{29}\) requires `reportlab` and `markdown`:

```bash
virtualenv -p /usr/bin/python2.7 /home/omero/omeroenv
source /home/omero/omeroenv/bin/activate
/home/omero/omeroenv/bin/pip2.7 install reportlab markdown
```

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 6.1.3 OMERO.server installation on CentOS 6 with Python 2.7 and Ice 3.6

This installation walkthrough should be read in conjunction with OMERO.server installation and OMERO.web deployment.

Running OMERO on CentOS 6 has a number of special requirements which deviate from the standard installation instructions. The instructions below will set up Python 2.7 and Ice 3.6 on CentOS 6. We tested the installation with Python 2.7 from IUS\(^{30}\) and used a virtual environment\(^{31}\) to install the various dependencies required to install an OMERO.server. It is also possible to use SCL Python (for example `walkthrough_centos6_py27.sh`) but such solution could have potential side effects.

**Setting up**

**Python 2.7**

CentOS 6 provides Python 2.6. However, OMERO.web requires Python 2.7 in order to use Django 1.8\(^{32}\). While Django 1.6 may be used with Python 2.6, this version of Django no longer has security support. In consequence, it is necessary to upgrade to Python 2.7 in order to obtain Django security updates, which are required for a production deployment.

**Ice 3.6**

With Ice 3.6, the Python bindings are provided separately. This allows to install the RPM packages provided by ZeroC for CentOS 6. Then run `pip install zeroc-ice` to install the Ice Python bindings if your package manager does not provide the Ice python packages. See Using the Python Distribution\(^{33}\) for further details.

---

\(^{28}\)http://wiki.centos.org/HowTos/SELinux

\(^{29}\)http://figure.openmicroscopy.org/

\(^{30}\)https://ius.io/

\(^{31}\)https://virtualenv.readthedocs.org/en/latest/

\(^{32}\)https://docs.djangoproject.com/en/1.8/releases/1.8/

\(^{33}\)https://doc.zeroc.com/display/Ice36/Using+the+Python+Distribution
Installing prerequisites

The following steps are run as root.

Install Java 1.8, Ice 3.6 and PostgreSQL 9.4:

To install Java 1.8 and other dependencies:

```
# epel-release will be pulled as a dependency
yum -y install https://centos6.iuscommunity.org/ius-release.rpm

# installed for convenience
yum -y install unzip wget tar

# install Java
yum -y install java-1.8.0-openjdk

# install dependencies
yum -y install \
  python27 \
  python27-devel \
  libjpeg-devel \
  libpng-devel \
  libtiff-devel \
  hdf5-devel \
  zlib-devel \
  freetype-devel

# install pip and virtualenv using Python 2.6
yum -y install python-pip

# if virtualenv is not installed (unlikely)
#yum -y install python27-pip
#pip2.7 install virtualenv

# TODO: this installs a lot of unnecessary packages:
yum -y groupinstall "Development Tools"

export PYTHONWARNINGS="ignore:Unverified HTTPS request"
```

To install Ice 3.6:

```
cd /etc/yum.repos.d
wget https://zeroc.com/download/rpm/zeroc-ice-el6.repo

yum -y install gcc-c++
yum -y install db53 db53-utils
yum -y install ice-all-runtime ice-all-devel

yum -y install openssl-devel bzip2-devel expat-devel

virtualenv -p /usr/bin/python2.7 /home/omero/omeroenv
set +u
source /home/omero/omeroenv/bin/activate
set -u

/home/omero/omeroenv/bin/pip2.7 install "zeroc-ice>3.5,<3.7"

deactivate
```

To install PostgreSQL 9.4:

6.1. Server installation on UNIX (and UNIX-like platforms)
The remaining dependencies will be installed in a virtual environment:

```
# Install the OMERO dependencies in a virtual environment
# -p only required if virtualenv has been installed with python 2.6
virtualenv -p /usr/bin/python2.7 /home/omero/omeroenv
set +u
source /home/omero/omeroenv/bin/activate
set -u
/home/omero/omeroenv/bin/pip install --upgrade pip
/home/omero/omeroenv/bin/pip2.7 install -r requirements_centos6_py27_ius.txt
deactivate
See requirements_centos6_py27_ius.txt
```

Create a database user and initialize a new database for OMERO:

```
echo "CREATE USER $OMERO_DB_USER PASSWORD '$OMERO_DB_PASS'" | \
   su - postgres -c psql
su - postgres -c "createdb -E UTF8 -O '$OMERO_DB_USER' '$OMERO_DB_NAME'"
psql -P pager=off -h localhost -U "$OMERO_DB_USER" -l
```

The following settings will need adding to your OMERO startup script or to the omero user’s environment (for example in a shell startup script). Add the absolute path to the bin directory of the virtual environment /home/omero/omeroenv to the PATH variable:

```
echo "export PATH="/"home/omero/omeroenv/bin:$PATH"" >> ~omero/.bashrc
```

These settings will enable Python 2.7, and set the necessary environment variables for Ice 3.6 to work.

**Install OMERO.server**

The following steps are run as the omero system user.

Download, unzip and configure OMERO. The rest of this walkthrough assumes the OMERO.server is installed into the home directory of the omero system user.
Note that this script requires the same environment variables that were set earlier in settings.env, so you may need to copy and/or source this file as the omero user.

You will need to install the server corresponding to your Ice version.

Install server-ice36.zip:

cd ~omero
SERVER=http://downloads.openmicroscopy.org/latest/omero5.2/server-ice36.zip
wget $SERVER -O OMERO.server-ice36.zip
unzip -q OMERO.server*

Configure:

ln -s OMERO.server-*/ OMERO.server
OMERO.server/bin/omero config set omero.data.dir "$OMERO_DATA_DIR"
OMERO.server/bin/omero config set omero.db.name "$OMERO_DB_NAME"
OMERO.server/bin/omero config set omero.db.user "$OMERO_DB_USER"
OMERO.server/bin/omero config set omero.db.pass "$OMERO_DB_PASS"
OMERO.server/bin/omero db script -f OMERO.server/db.sql --password "$OMERO_ROOT_PASS"
psql -h localhost -U "$OMERO_DB_USER" "$OMERO_DB_NAME" < OMERO.server/db.sql

Installing a web server

To Deploy OMERO.web, you can use either Nginx or Apache. Follow the steps to install your chosen web server.

Nginx

See also OMERO.web Nginx and Gunicorn deployment (Unix/Linux).

The following steps are run as root.

Install Nginx 1.8, install the requirements to run OMERO.web in the virtual environment, deactivate it and copy the Nginx OMERO configuration file into the Nginx configuration directory, and disable the default configuration:

cat << EOF > /etc/yum.repos.d/nginx.repo
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=0
enabled=1
EOF

#install nginx
yum -y install nginx

virtualenv -p /usr/bin/python2.7 /home/omero/omeroenv
set +u
source /home/omero/omeroenv/bin/activate
set -u

# Install OMERO.web requirements
/home/omero/omeroenv/bin/pip2.7 install -r ~omero/OMERO.server/share/web/requirements-py27-nginx.txt
deactivate

# set up as the omero user.
su - omero -c "bash -euX setup_omero_nginx.sh"

mv /etc/nginx/conf.d/default.conf /etc/nginx/conf.d/default.disabled

6.1. Server installation on UNIX (and UNIX-like platforms)
cp ~omero/OMERO.server/nginx.conf.tmp /etc/nginx/conf.d/omero-web.conf

service nginx start

Apache

See also DEPRECATED: OMERO.web Apache and mod_wsgi deployment (Unix/Linux).

As the omero system user, configure OMERO.web by running the following commands:

OMERO.server/bin/omero config set omero.web.application_server wsgi
OMERO.server/bin/omero web config apache --http "$OMERO_WEB_PORT" > OMERO.server/apache.conf.tmp
OMERO.server/bin/omero web syncmedia

The following steps are run as root.

The version of Apache installed is 2.2, see the note below if you wish to use version 2.4.

Install Apache 2.2, install the requirements to run OMERO.web in the virtual environment, deactivate it and add the path to the virtual environment python to the python-path parameter of the WSGIDaemonProcess directive:

#install Apache 2.2
yum -y install httpd

# install mod_wsgi compiled against 2.7
yum -y install python27-mod_wsgi

virtualenv -p /usr/bin/python2.7 /home/omero/omeroenv
set +u
source /home/omero/omeroenv/bin/activate
set -u

# Install OMERO.web requirements
/home/omero/omeroenv/bin/pip2.7 install -r ~omero/OMERO.server/share/web/requirements-py27-apache.txt
deactivate

# Add virtual env python to the python-path parameter of the WSGIDaemonProcess directive
sed -i 's/\(python-path\=\)/\1/home/omero/omeroenv/\lib64\/python2.7\/site-packages//' ~/omero/OMERO.web/apache.conf.tmp

chkconfig httpd on

service httpd start

Note: If you wish to use Apache 2.4 with mod_wsgi, see omero-install\(^{34}\) for a possible solution.

Running OMERO

The following steps are run as the omero system user.

OMERO should now be set up. To start the server run:

OMERO.server/bin/omero admin start

\(^{34}\)https://github.com/ome/omero-install

6.1. Server installation on UNIX (and UNIX-like platforms)
If you deploy with Nginx, to start the OMERO.web client run:

OMERO.server/bin/omero web start

The last command is not necessary if you deploy with Apache.
Nginx or Apache should already be running so you should be able to log in as the OMERO root user by going to http://localhost/ in your web browser. Please read the SELinux section below.

In addition some example init.d scripts are available should you wish to start OMERO and OMERO.web automatically:

cp omero-init.d /etc/init.d/omero
chmod a+x /etc/init.d/omero

cp omero-web-init.d /etc/init.d/omero-web
chmod a+x /etc/init.d/omero-web

chkconfig --del omero
chkconfig --add omero
chkconfig --del omero-web
chkconfig --add omero-web

omero-init.d
omero-web-init.d

**Securing OMERO**

**The following steps are run as root.**

If multiple users have access to the machine running OMERO you should restrict access to OMERO.server’s configuration and runtime directories, and optionally the OMERO data directory:

chmod go-rwx ~omero/OMERO.server/etc ~omero/OMERO.server/var

# Optionally restrict access to the OMERO data directory
#chomd go-rwx "$OMERO_DATA_DIR"

**Regular tasks**

**The following steps are run as root.**

The default OMERO.web session handler uses temporary files to store sessions which should be deleted at regular intervals, for instance by creating a cron job:

OMERO_USER=omero
OMERO_SERVER=/home/omero/OMERO.server
su - ${OMERO_USER} -c "${OMERO_SERVER}/bin/omero web clearsessions"

Copy the following commands into the appropriate location:

cp omero-web-cron /etc/cron.daily/omero-web
chmod a+x /etc/cron.daily/omero-web

omero-web-cron
**OMERO Documentation, Release 5.2.6**

**SELinux**

The following steps are run as root.

If you are running a system with SELinux enabled (it is enabled by default on CentOS 6\(^{35}\)) and are unable to access OMERO.web you may need to adjust the security policy:

```bash
if [ $(getenforce) != Disabled ]; then
    yum -y install policycoreutils-python
    setsebool -P httpd_read_user_content 1
    setsebool -P httpd_enable_homedirs 1
    semanage port =a -t http_port_t -p tcp 4080
fi
```

**Installing Web apps**

The following steps are run as root.

It is possible to add Web applications to OMERO. If your app required some extra Python packages installed using pip, those packages should be also installed in the virtual environment. For example, OMERO.figure\(^{36}\) requires reportlab and markdown:

```
virtualenv -p /usr/bin/python2.7 /home/omero/omeroenv
source /home/omero/omeroenv/bin/activate
/home/omero/omeroenv/bin/pip2.7 install reportlab markdown
```

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 6.1.4 OMERO.server installation on CentOS 7 and Ice 3.5

This is an example walkthrough for installing OMERO on CentOS 7, using a dedicated system user, and should be read in conjunction with OMERO.web deployment. You can use this as a guide for setting up your own test server. For production use you should also read the pages listed under Optimizing Server Configuration.

These instructions assume your Linux distribution is configured with a UTF-8 locale (this is normally the default).

For convenience in this walkthrough the main OMERO configuration options have been defined as environment variables. When following this walkthrough you can either use your own values, or alternatively source the following file:

```
OMERO_DB_USER=db_user
OMERO_DB_PASS=db_password
OMERO_DB_NAME=omero_database
OMERO_ROOT_PASS=omero_root_password
OMERO_DATA_DIR=/OMERO
OMERO_WEB_PORT=80

export OMERO_DB_USER OMERO_DB_PASS OMERO_DB_NAME OMERO_ROOT_PASS OMERO_DATA_DIR OMERO_WEB_PORT
export PGPASSWORD="$OMERO_DB_PASS"
```

settings.env

\(^{35}\)http://wiki.centos.org/How_Tos/SELinux

\(^{36}\)http://figure.openmicroscopy.org/
Installing prerequisites

The following steps are run as root.

Install Java 1.8, Ice 3.5 and PostgreSQL 9.4:

To install Java 1.8 and other dependencies:

```
yum -y install epel-release
# installed for convenience
yum -y install unzip wget

# install Java
yum -y install java-1.8.0-openjdk
# install dependencies
yum -y install \
    python-pip python-devel python-virtualenv \n    numpy scipy python-matplotlib python-tables
yum -y install \
    zlib-devel \n    libjpeg-devel \n    gcc
pip install --upgrade pip
pip install -r 'dirname $0'/requirements.txt
```

To install Ice 3.5:

```
curl -o /etc/yum.repos.d/zeroc-ice-el7.repo \n    http://download.zeroc.com/Ice/3.5/el7/zeroc-ice-el7.repo
yum -y install ice ice-python ice-servers
```

To install PostgreSQL 9.4:

```
# install Postgres
# Postgres, reconfigure to allow TCP connections
yum -y install http://yum.postgresql.org/9.4/redhat/rhel-7-x86_64/pgdg-centos94-9.4-2.noarch.rpm
yum -y install postgresql94-server postgresql94
PGSETUP_INITDB_OPTIONS=--encoding=UTF8 /usr/bin/postgresql94-setup initdb
sed -i.bak -re 's/^ident/\1md5/ /usr/lib64/pgsql/9.4/data/pg_hba.conf
systemctl start postgresql-9.4.service
systemctl enable postgresql-9.4.service
```

See requirements.txt

Create an omero system user, and a directory for the OMERO repository:

```
useradd -m omero
chmod a+X ~omero
mkdir -p "$OMERO_DATA_DIR"
chown omero "$OMERO_DATA_DIR"
```

Create a database user and initialize a new database for OMERO:
echo "CREATE USER $OMERO_DB_USER PASSWORD '$OMERO_DB_PASS'" | \
  su - postgres -c psql
su - postgres -c "createdb -E UTF8 -O '$OMERO_DB_USER' '$OMERO_DB_NAME'"
psql -P pager=off -h localhost -U "$OMERO_DB_USER" -l

Installing OMERO.server

The following steps are run as the omero system user.

Download, unzip and configure OMERO. The rest of this walkthrough assumes the OMERO.server is installed into the home directory of the omero system user.

Note that this script requires the same environment variables that were set earlier in settings.env, so you may need to copy and/or source this file as the omero user.

You will need to install the server corresponding to your Ice version.

Install server-ice35.zip:

cd ~omero
SERVER=http://downloads.openmicroscopy.org/latest/omero5.2/server-ice35.zip
wget $SERVER
unzip -q OMERO.server*

Configure:

ln -s OMERO.server*/ OMERO.server
OMERO.server/bin/omero config set omero.data.dir "$OMERO_DATA_DIR"
OMERO.server/bin/omero config set omero.db.name "$OMERO_DB_NAME"
OMERO.server/bin/omero config set omero.db.user "$OMERO_DB_USER"
OMERO.server/bin/omero config set omero.db.pass "$OMERO_DB_PASS"
OMERO.server/bin/omero db script -f OMERO.server/db.sql --password "$OMERO_ROOT_PASS"
psql -h localhost -U "$OMERO_DB_USER" "$OMERO_DB_NAME" < OMERO.server/db.sql

Installing a web server

To Deploy OMERO.web, you can use either Nginx or Apache. Follow the steps to install your chosen web server.

Nginx

See also OMERO.web Nginx and Gunicorn deployment (Unix/Linux).

The following steps are run as root.

Install Nginx 1.8, copy the Nginx OMERO configuration file into the Nginx configuration directory, and disable the default configuration:

# The following is only required to install
# latest stable version of nginx
# Default will be 1.6.3 if not set
cat << EOF > /etc/yum.repos.d/nginx.repo
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=0
enabled=1
EOF

#install nginx
yum -y install nginx

pip install -r ~omero/OMERO.server/share/web/requirements-py27-nginx.txt

# set up as the omero user.
su - omero -c "bash -eux setup_omero_nginx.sh"

sed -i.bak -re 's/( default_server.*)/; #\1/ /etc/nginx/nginx.conf
mv /etc/nginx/conf.d/default.conf /etc/nginx/conf.d/default.disabled
cp ~omero/OMERO.server/nginx.conf.tmp /etc/nginx/conf.d/omero-web.conf

systemctl enable nginx

systemctl start nginx

Apache

See also DEPRECATED: OMERO.web Apache and mod_wsgi deployment (Unix/Linux).

As the omero system user, configure OMERO.web:

OMERO.server/bin/omero config set omero.web.application_server wsgi
OMERO.server/bin/omero web config apache24 --http "$OMERO_WEB_PORT" > OMERO.server/apache.conf.tmp
OMERO.server/bin/omero web syncmedia

The following steps are run as root.

Install Apache 2.4 and copy the Apache OMERO configuration file into the Apache configuration directory:

yum -y install httpd mod_wsgi

# Install OMERO.web requirements
pip install -r ~omero/OMERO.server/share/web/requirements-py27-apache.txt

cp ~omero/OMERO.server/apache.conf.tmp /etc/httpd/conf.d/omero-web.conf

cp -rf /run/httpd/* /tmp/httpd*

systemctl enable httpd.service

systemctl start httpd

Running OMERO

The following steps are run as the omero system user.

OMERO should now be set up. To start the server run:

OMERO.server/bin/omero admin start

If you deploy with Nginx, to start the OMERO.web client run:
OMERO.server/bin/omero web start

The last command is not necessary if you deploy with Apache.
Nginx or Apache should already be running so you should be able to log in as the OMERO root user by going to http://localhost/ in your web browser. Please read the SELinux section below.

In addition some example systemd.service scripts are available should you wish to start OMERO and OMERO.web automatically:

```bash
cp omero-systemd.service /etc/systemd/system/omero.service
cp omero-web-systemd.service /etc/systemd/system/omero-web.service
systemctl daemon-reload
systemctl enable omero.service
systemctl enable omero-web.service
```

```
omeno-systemd.service
omero-web-systemd.service
```

**Securing OMERO**

**The following steps are run as root.**

If multiple users have access to the machine running OMERO you should restrict access to OMERO.server’s configuration and runtime directories, and optionally the OMERO data directory:

```bash
chmod go-rwx ~omero/OMERO.server/etc ~omero/OMERO.server/var
```

```
# Optionally restrict access to the OMERO data directory
chmod go-rwx "$OMERO_DATA_DIR"
```

**Regular tasks**

**The following steps are run as root.**

The default OMERO.web session handler uses temporary files to store sessions which should be deleted at regular intervals, for instance by creating a cron job:

```bash
OMERO_USER=omero
OMERO_SERVER=/home/omero/OMERO.server
su - ${OMERO_USER} -c "${OMERO_SERVER}/bin/omero web clearsessions"
```

Copy this script into the appropriate location:

```bash
cp omero-web-cron /etc/cron.daily/omero-web
chmod a+x /etc/cron.daily/omero-web
```

```
omeno-web-cron
```

6.1. Server installation on UNIX (and UNIX-like platforms)
**SELinux**

The following steps are run as root.

If you are running a system with SELinux enabled\(^\text{[37]}\) and are unable to access OMERO.web you may need to adjust the security policy:

```bash
if [ $(getenforce) != Disabled ]; then
    yum -y install policycoreutils-python
    setsebool -P httpd_read_user_content 1
    setsebool -P httpd_enable_homedirs 1
    semanage port =a -t http_port_t -p tcp 4080
fi
```

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

---

### 6.1.5 OMERO.server installation on CentOS 7 and Ice 3.6

This is an example walkthrough for installing OMERO on CentOS 7, using a dedicated system user, and should be read in conjunction with OMERO.web deployment. You can use this as a guide for setting up your own test server. For production use you should also read the pages listed under Optimizing Server Configuration.

These instructions assume your Linux distribution is configured with a UTF-8 locale (this is normally the default).

For convenience in this walkthrough the main OMERO configuration options have been defined as environment variables. When following this walkthrough you can either use your own values, or alternatively source the following file:

```bash
OMERO_DB_USER=db_user
OMERO_DB_PASS=db_password
OMERO_DB_NAME=omero_database
OMERO_ROOT_PASS=omero_root_password
OMERO_DATA_DIR=/OMERO
OMERO_WEB_PORT=80

export OMERO_DB_USER OMERO_DB_PASS OMERO_DB_NAME OMERO_ROOT_PASS OMERO_DATA_DIR OMERO_WEB_PORT
export PGPASSWORD="$OMERO_DB_PASS"

settings.env
```

**Installing prerequisites**

The following steps are run as root.

Install Java 1.8, Ice 3.6 and PostgreSQL 9.4:

To install Java 1.8 and other dependencies:

```bash
yum -y install epel-release
# installed for convenience
yum -y install unzip wget
# install Java
yum -y install java-1.8.0-openjdk
```

---

\(^{[37]}\)http://wiki.centos.org/HowTos/SELinux

---

6.1. Server installation on UNIX (and UNIX-like platforms) 112
OMERO Documentation, Release 5.2.6

# install dependencies
yum -y install \
python-pip python-devel python-virtualenv \
numpy scipy python-matplotlib python-tables
yum -y install \
zlib-devel \
libjpeg-devel \
gcc
pip install --upgrade pip
pip install -r ‘dirname $0‘/requirements.txt

To install Ice 3.6:
cd /etc/yum.repos.d
wget https://zeroc.com/download/rpm/zeroc-ice-el7.repo
yum -y install gcc-c++
yum -y install libdb-utils
yum -y install openssl-devel bzip2-devel expat-devel
yum -y install ice-all-runtime ice-all-devel
pip install ”zeroc-ice>3.5,<3.7”

To install PostgreSQL 9.4:
# install Postgres
# Postgres, reconﬁgure to allow TCP connections
yum -y install http://yum.postgresql.org/9.4/redhat/rhel-7-x86_64/pgdg-centos94-9.4-2.noarch.rpm
yum -y install postgresql94-server postgresql94
PGSETUP_INITDB_OPTIONS=--encoding=UTF8 /usr/pgsql-9.4/bin/postgresql94-setup initdb
sed -i.bak -re ’s/^(host.*)ident/\1md5/’ /var/lib/pgsql/9.4/data/pg_hba.conf
systemctl start postgresql-9.4.service
systemctl enable postgresql-9.4.service

See requirements.txt
Create an omero system user, and a directory for the OMERO repository:
useradd -m omero
chmod a+X ~omero
mkdir -p ”$OMERO_DATA_DIR”
chown omero ”$OMERO_DATA_DIR”

Create a database user and initialize a new database for OMERO:
echo ”CREATE USER $OMERO_DB_USER PASSWORD ’$OMERO_DB_PASS’” | \
su - postgres -c psql
su - postgres -c ”createdb -E UTF8 -O ’$OMERO_DB_USER’ ’$OMERO_DB_NAME’”
psql -P pager=oﬀ -h localhost -U ”$OMERO_DB_USER” -l

6.1. Server installation on UNIX (and UNIX-like platforms)

113


Installing OMERO.server

The following steps are run as the omero system user.

Download, unzip and configure OMERO. The rest of this walkthrough assumes the OMERO.server is installed into the home directory of the omero system user.

Note that this script requires the same environment variables that were set earlier in settings.env, so you may need to copy and/or source this file as the omero user.

You will need to install the server corresponding to your Ice version.

Install server-ice36.zip:

```
cd ~omero
SERVER=http://downloads.openmicroscopy.org/latest/omero5.2/server-ice36.zip
wget $SERVER -O OMERO.server-ice36.zip
unzip -q OMERO.server
```

Configure:

```
ln -s OMERO.server-*/ OMERO.server
OMERO.server/bin/omero config set omero.data.dir "$OMERO_DATA_DIR"
OMERO.server/bin/omero config set omero.db.name "$OMERO_DB_NAME"
OMERO.server/bin/omero config set omero.db.user "$OMERO_DB_USER"
OMERO.server/bin/omero config set omero.db.pass "$OMERO_DB_PASS"
OMERO.server/bin/omero db script -f OMERO.server/db.sql --password "$OMERO_ROOT_PASS"
psql -h localhost -U "$OMERO_DB_USER" "$OMERO_DB_NAME" < OMERO.server/db.sql
```

Installing a web server

To Deploy OMERO.web, you can use either Nginx or Apache. Follow the steps to install your chosen web server.

Nginx

See also OMERO.web Nginx and Gunicorn deployment (Unix/Linux).

The following steps are run as root.

Install Nginx 1.8, copy the Nginx OMERO configuration file into the Nginx configuration directory, and disable the default configuration:

```
# The following is only required to install
# latest stable version of nginx
# Default will be 1.6.3 if not set
cat << EOF > /etc/yum.repos.d/nginx.repo
[nginx]
nname=nginx repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=0
enabled=1
EOF

#install nginx
yum -y install nginx

pip install -r ~omero/OMERO.server/share/web/requirements-py27-nginx.txt

# set up as the omero user.
su - omero -c "bash -eux setup_omero_nginx.sh"
```
sed -i.bak -re 's/( default_server.*)/; #\1/' /etc/nginx/nginx.conf
mv /etc/nginx/conf.d/default.conf /etc/nginx/conf.d/default.disabled
cp ~omero/OMERO.server/nginx.conf.tmp /etc/nginx/conf.d/omero-web.conf

systemctl enable nginx
systemctl start nginx

Apache

See also DEPRECATED: OMERO.web Apache and mod_wsgi deployment (Unix/Linux).

As the omero system user, configure OMERO.web:

OMERO.server/bin/omero config set omero.web.application_server wsgi
OMERO.server/bin/omero web config apache24 --http "$OMERO_WEB_PORT" > OMERO.server/apache.conf.tmp
OMERO.server/bin/omero web syncmedia

The following steps are run as root.

Install Apache 2.4 and copy the Apache OMERO configuration file into the Apache configuration directory:

yum -y install httpd mod_wsgi
# Install OMERO.web requirements
pip install -r ~omero/OMERO.server/share/web/requirements-py27-apache.txt
cp ~omero/OMERO.server/apache.conf.tmp /etc/httpd/conf.d/omero-web.conf
rm -rf /run/httpd/* /tmp/httpd*
systemctl enable httpd.service
systemctl start httpd

Running OMERO

The following steps are run as the omero system user.

OMERO should now be set up. To start the server run:

OMERO.server/bin/omero admin start

If you deploy with Nginx, to start the OMERO.web client run:

OMERO.server/bin/omero web start

The last command is not necessary if you deploy with Apache.

Nginx or Apache should already be running so you should be able to log in as the OMERO root user by going to http://localhost/ in your web browser. Please read the SELinux section below.

In addition some example systemd.service scripts are available should you wish to start OMERO and OMERO.web automatically:
securing OMERO

The following steps are run as root.

If multiple users have access to the machine running OMERO you should restrict access to OMERO.server’s configuration and runtime directories, and optionally the OMERO data directory:

```
chmod go-rwx ~omero/OMERO.server/etc ~omero/OMERO.server/var
```

# Optionally restrict access to the OMERO data directory
# chmod go-rwx "$OMERO_DATA_DIR"

regular tasks

The following steps are run as root.

The default OMERO.web session handler uses temporary files to store sessions which should be deleted at regular intervals, for instance by creating a cron job:

```
OMERO_USER=omero
OMERO_SERVER=/home/omero/OMERO.server
su - ${OMERO_USER} -c "/home/omero/OMERO.server/bin/omero web clearsessions"
```

Copy this script into the appropriate location:

```
cp omero-web-cron /etc/cron.daily/omero-web
chmod a+x /etc/cron.daily/omero-web
```

SELinux

The following steps are run as root.

If you are running a system with SELinux enabled\(^{38}\) and are unable to access OMERO.web you may need to adjust the security policy:

\(^{38}\)http://wiki.centos.org/HowTos/SELinux
if [ $(getenforce) != Disabled ]; then
    yum -y install policycoreutils-python
    setsebool -P httpd_read_user_content 1
    setsebool -P httpd_enable_homedirs 1
    semanage port -a -t http_port_t -p tcp 4080
fi

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.1.6 OMERO.server installation on Ubuntu 14.04 and Ice 3.5

This is an example walkthrough for installing OMERO on Ubuntu, using a dedicated system user, and should be read in conjunction with OMERO.web deployment. You can use this as a guide for setting up your own test server. For production use you should also read the pages listed under Optimizing Server Configuration.

These instructions assume your Linux distribution is configured with a UTF-8 locale (this is normally the default).

For convenience in this walkthrough the main OMERO configuration options have been defined as environment variables. When following this walkthrough you can either use your own values, or alternatively source the following file:

OMERO_DB_USER=db_user
OMERO_DB_PASS=db_password
OMERO_DB_NAME=omero_database
OMERO_ROOT_PASS=omero_root_password
OMERO_DATA_DIR=/OMERO
OMERO_WEB_PORT=80

export OMERO_DB_USER OMERO_DB_PASS OMERO_DB_NAME OMERO_ROOT_PASS OMERO_DATA_DIR OMERO_WEB_PORT

export PGPASSWORD="$OMERO_DB_PASS"

settings.env

Installing prerequisites

The following steps are run as root.

Install Java 1.8, Ice 3.5 and PostgreSQL 9.4:

To install Java 1.8 and other dependencies:

apt-get update

# installed for convenience
apt-get -y install unzip wget

# install Java
apt-get -y install software-properties-common
add-apt-repository -y ppa:openjdk-r/ppa
apt-get update
apt-get -y install openjdk-8-jre

# install dependencies
apt-get update
apt-get -y install \
    unzip \

6.1. Server installation on UNIX (and UNIX-like platforms)
wget \
python-(matplotlib,numpy,pip,scipy,tables,virtualenv)

# require to install Pillow
apt-get -y install \
  libtiff5-dev \ 
  libjpeg8-dev \ 
  zlib1g-dev \ 
  libfreetype6-dev \ 
  liblcms2-dev \ 
  libwebp-dev \ 
  tcl8.6-dev \ 
  tk8.6-dev

pip install --upgrade pip

# upgrade required since pillow is already installed
pip install --upgrade -r requirements.txt

To install Ice 3.5:

apt-get -y install ice-services python-zeroc-ice

To install PostgreSQL 9.4:

# install Postgres
apt-get -y install apt-transport-https
add-apt-repository -y "deb https://apt.postgresql.org/pub/repos/apt/ trusty-pgdg main" 
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | apt-key add - 
apt-get update 
apt-get -y install postgresql-9.4 
service postgresql start

See requirements.txt

Create an omero system user, and a directory for the OMERO repository:

useradd -m omero 
chmod a+X ~omero 
mkdir -p "$OMERO_DATA_DIR" 
chown omero "$OMERO_DATA_DIR"

Create a database user and initialize a new database for OMERO:

echo "CREATE USER $OMERO_DB_USER PASSWORD '$OMERO_DB_PASS'" | \
su - postgres -c psql 
su - postgres -c "createdb -E UTF8 -O '$OMERO_DB_USER' '$OMERO_DB_NAME'" 
psql -P pager=off -h localhost -U "$OMERO_DB_USER" -l

Installing OMERO.server

The following steps are run as the omero system user.

Download, unzip and configure OMERO. The rest of this walkthrough assumes the OMERO.server is installed into the home directory of the omero system user.
Note that this script requires the same environment variables that were set earlier in settings.env, so you may need to copy and/or source this file as the omero user.

You will need to install the server corresponding to your Ice version.

Install server-ice35.zip:

```bash
cd ~omero
SERVER=http://downloads.openmicroscopy.org/latest/omero5.2/server-ice35.zip
wget $SERVER
unzip -q OMERO.server*
```

Configure:

```bash
ln -s OMERO.server-*/* OMERO.server
OMERO.server/bin/omero config set omero.data.dir "$OMERO_DATA_DIR"
OMERO.server/bin/omero config set omero.db.name "$OMERO_DB_NAME"
OMERO.server/bin/omero config set omero.db.user "$OMERO_DB_USER"
OMERO.server/bin/omero config set omero.db.pass "$OMERO_DB_PASS"
OMERO.server/bin/omero db script -f OMERO.server/db.sql --password "$OMERO_ROOT_PASS"
psql -h localhost -U "$OMERO_DB_USER" "$OMERO_DB_NAME" < OMERO.server/db.sql
```

### Installing a web server

To Deploy OMERO.web, you can use either Nginx or Apache. Follow the steps to install your chosen web server.

**Nginx**

See also [OMERO.web Nginx and Gunicorn deployment (Unix/Linux)](#).

The following steps are run as root.

Install Nginx 1.8, copy the Nginx OMERO configuration file into the Nginx configuration directory, and disable the default configuration:

```bash
# require to install more recent version of nginx
# w/o the version installed is 1.4.6
add-apt-repository -y ppa:nginx/stable
apt-get update
apt-get -y install nginx
pip install -r ~omero/OMERO.server/share/web/requirements-py27-nginx.txt
```

```bash
# set up as the omero user.
su - omero -c "bash -eux setup_omero_nginx.sh"
```

```bash
cp ~omero/OMERO.server/nginx.conf.tmp /etc/nginx/sites-available/omero-web
rm /etc/nginx/sites-enabled/default
ln -s /etc/nginx/sites-available/omero-web /etc/nginx/sites-enabled/
```

```
service nginx start
```

**Apache**

See also [DEPRECATED: OMERO.web Apache and mod_wsgi deployment (Unix/Linux)](#).

As the omero system user, configure OMERO.web:

```
6.1. Server installation on UNIX (and UNIX-like platforms) 119
```
OMERO.server/bin/omero config set omero.web.application_server wsgi
OMERO.server/bin/omero web config apache24 --http "${OMERO_WEB_PORT}" > OMERO.server/apache.conf.tmp
OMERO.server/bin/omero web syncmedia

The following steps are run as root.

Install Apache 2.4 and copy the Apache OMERO configuration file into the Apache configuration directory, and disable the default configuration:

apt-get -y install apache2 libapache2-mod-wsgi

# Install OMERO.web requirements
pip install -r ~omero/OMERO.server/share/web/requirements-py27-apache.txt

# Modify the default value set for the 'WSGISocketPrefix' directive in 'apache.conf.tmp'
  sed -i -r -e 's|\(WSGISocketPrefix run/wsgi\)|#\1|' -e 's|# \(WSGISocketPrefix /var/run/wsgi\)|\1|' ~omero/OMERO.server/apache.conf.tmp
  cp ~omero/OMERO.server/apache.conf.tmp /etc/apache2/sites-available/omero-web.conf
  a2dissite 000-default.conf
  a2ensite omero-web.conf

  service apache2 start

Running OMERO

The following steps are run as the omero system user.

OMERO should now be set up. To start the server run:

OMERO.server/bin/omero admin start

If you deploy with Nginx, to start the OMERO.web client run:

OMERO.server/bin/omero web start

The last command is not necessary if you deploy with Apache.

Nginx or Apache should already be running so you should be able to log in as the OMERO root user by going to http://localhost/ in your web browser.

In addition some example init.d scripts are available should you wish to start OMERO and OMERO.web automatically:

cp omero-init.d /etc/init.d/omero
chmod a+x /etc/init.d/omero

cp omero-web-init.d /etc/init.d/omero-web
chmod a+x /etc/init.d/omero-web

update-rc.d -f omero remove
update-rc.d -f omero defaults 98 02
update-rc.d -f omero-web remove
update-rc.d -f omero-web defaults 98 02

omero-init.d
omero-web-init.d

6.1. Server installation on UNIX (and UNIX-like platforms)
Securing OMERO

The following steps are run as root.

If multiple users have access to the machine running OMERO you should restrict access to OMERO.server’s configuration and runtime directories, and optionally the OMERO data directory:

```bash
chmod go-rwx ~omero/OMERO.server/etc ~omero/OMERO.server/var
# Optionally restrict access to the OMERO data directory
#chmod go-rwx "$OMERO_DATA_DIR"
```

Regular tasks

The following steps are run as root.

The default OMERO.web session handler uses temporary files to store sessions which should be deleted at regular intervals, for instance by creating a cron job:

```bash
OMERO_USER=omero
OMERO_SERVER=/home/omero/OMERO.server
su - ${OMERO_USER} -c "${OMERO_SERVER}/bin/omero web clearsessions"
```

Copy the following commands into the appropriate location:

```bash
cp omero-web-cron /etc/cron.daily/omero-web
chmod a+x /etc/cron.daily/omero-web
```

OMERO-web-cron

---

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.1.7 OMERO.server installation on Ubuntu 14.04 and Ice 3.6

This is an example walkthrough for installing OMERO on Ubuntu, using a dedicated system user, and should be read in conjunction with OMERO.web deployment. You can use this as a guide for setting up your own test server. For production use you should also read the pages listed under Optimizing Server Configuration.

These instructions assume your Linux distribution is configured with a UTF-8 locale (this is normally the default).

For convenience in this walkthrough the main OMERO configuration options have been defined as environment variables. When following this walkthrough you can either use your own values, or alternatively source the following file:

```bash
OMERO_DB_USER=db_user
OMERO_DB_PASS=db_password
OMERO_DB_NAME=omero_database
OMERO_ROOT_PASS=omero_root_password
OMERO_DATA_DIR=/OMERO
OMERO_WEB_PORT=80
export OMERO_DB_USER OMERO_DB_PASS OMERO_DB_NAME OMERO_ROOT_PASS OMERO_DATA_DIR OMERO_WEB_PORT
```
export PGPASSWORD="$OMERO_DB_PASS"

settings.env

Installing prerequisites

The following steps are run as root.
Install Java 1.8, Ice 3.6 and PostgreSQL 9.4:
To install Java 1.8 and other dependencies:

```bash
apt-get update
# installed for convenience
apt-get -y install unzip wget
# install Java
apt-get -y install software-properties-common
add-apt-repository -y ppa:openjdk-r/ppa
apt-get update
apt-get -y install openjdk-8-jre
# install dependencies
apt-get update
apt-get -y install \
  unzip \
  wget \
  python-{matplotlib,numpy,pip,scipy,tables,virtualenv}
# require to install Pillow
apt-get -y install \
  libtiff5-dev \
  libjpeg8-dev \
  zlib1g-dev \
  libfreetype6-dev \
  libxml2-dev \
  libwebp-dev \
  tcl8.6-dev \
  tk8.6-dev

pip install --upgrade pip
# upgrade required since pillow is already installed
pip install --upgrade -r requirements.txt
```

To install Ice 3.6:

```bash
apt-get -y install db5.3-util
apt-get -y install libssl-dev libbz2-dev libmcpp-dev libdb++-dev libdb-dev
apt-key adv --keyserver keyserver.ubuntu.com --recv 5E6DA83306132997
apt-add-repository "deb http://zeroc.com/download/apt/ubuntu'lsb_release -rs' stable main"
apt-get update
apt-get -y install zeroc-ice-all-runtime zeroc-ice-all-dev
pip install "zeroc-ice>3.5,<3.7"
```

To install PostgreSQL 9.4:
# install Postgres
apt-get -y install apt-transport-https
add-apt-repository -y "deb https://apt.postgresql.org/pub/repos/apt/ trusty-pgdg main"
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | apt-key add -
apt-get update
apt-get -y install postgresql-9.4
service postgresql start

See requirements.txt

Create an omero system user, and a directory for the OMERO repository:

useradd -m omero
chmod a+X ~omero

mkdir -p "$OMERO_DATA_DIR"
chown omero "$OMERO_DATA_DIR"

Create a database user and initialize a new database for OMERO:

echo "CREATE USER $OMERO_DB_USER PASSWORD '$OMERO_DB_PASS'" | \ 
su - postgres -c psql
su - postgres -c "createdb -E UTF8 -O '$OMERO_DB_USER' '$OMERO_DB_NAME'"
psql -P pager=off -h localhost -U "$OMERO_DB_USER" -l

Installing OMERO.server

The following steps are run as the omero system user.

Download, unzip and configure OMERO. The rest of this walkthrough assumes the OMERO.server is installed into the home directory of the omero system user.

Note that this script requires the same environment variables that were set earlier in settings.env, so you may need to copy and/or source this file as the omero user.

You will need to install the server corresponding to your Ice version.
Install server-ice36.zip:

cd ~omero
SERVER=http://downloads.openmicroscopy.org/latest/omero5.2/server-ice36.zip
wget $SERVER -O OMERO.server-ice36.zip
unzip -q OMERO.server*

Configure:

ln -s OMERO.server-*/ OMERO.server
OMERO.server/bin/omero config set omero.data.dir "$OMERO_DATA_DIR"
OMERO.server/bin/omero config set omero.db.name "$OMERO_DB_NAME"
OMERO.server/bin/omero config set omero.db.user "$OMERO_DB_USER"
OMERO.server/bin/omero config set omero.db.pass "$OMERO_DB_PASS"
OMERO.server/bin/omero db script -f OMERO.server/db.sql --password "$OMERO_ROOT_PASS"
psql -h localhost -U "$OMERO_DB_USER" "$OMERO_DB_NAME" < OMERO.server/db.sql

Installing a web server

To Deploy OMERO.web, you can use either Nginx or Apache. Follow the steps to install your chosen web server.

6.1. Server installation on UNIX (and UNIX-like platforms)
Nginx

See also OMERO.web Nginx and Gunicorn deployment (Unix/Linux).

The following steps are run as root.

Install Nginx 1.8, copy the Nginx OMERO configuration file into the Nginx configuration directory, and disable the default configuration:

```bash
# require to install more recent version of nginx
# w/o the version installed is 1.4.6
add-apt-repository -y ppa:nginx/stable
apt-get update
apt-get -y install nginx
pip install -r ~/omero/OMERO.server/share/web/requirements-py27-nginx.txt

# set up as the omero user.
su - omero -c "bash -eux setup_omero_nginx.sh"

cp ~/omero/OMERO.server/nginx.conf.tmp /etc/nginx/sites-available/omero-web
rm /etc/nginx/sites-enabled/default
ln -s /etc/nginx/sites-available/omero-web /etc/nginx/sites-enabled/

service nginx start
```

Apache

See also DEPRECATED: OMERO.web Apache and mod_wsgi deployment (Unix/Linux).

As the omero system user, configure OMERO.web:

```bash
OMERO.server/bin/omero config set omero.web.application_server wsgi
OMERO.server/bin/omero web config apache24 --http "$OMERO_WEB_PORT" > OMERO.server/apache.conf.tmp
OMERO.server/bin/omero web syncmedia

The following steps are run as root.

Install Apache 2.4 and copy the Apache OMERO configuration file into the Apache configuration directory, and disable the default configuration:

```bash
apt-get -y install apache2 libapache2-mod-wsgi

# Install OMERO.web requirements
pip install -r ~/omero/OMERO.server/share/web/requirements-py27-apache.txt

# Modify the default value set for the ''WSGISocketPrefix'' directive in ''apache.conf.tmp''
```

```bash
sed -i -e 's|\(WSGISocketPrefix run/wsgi\)|#\1|' -e 's|# \(WSGISocketPrefix /var/run/wsgi\)|\1|' ~/omero/OMERO.server/apache.conf.tmp
```

```bash
a2dissite 000-default.conf
a2ensite omero-web.conf

service apache2 start
```

Running OMERO

The following steps are run as the omero system user.

OMERO should now be set up. To start the server run:
OMERO.server/bin/omero admin start

If you deploy with Nginx, to start the OMERO.web client run:

OMERO.server/bin/omero web start

The last command is not necessary if you deploy with Apache.
Nginx or Apache should already be running so you should be able to log in as the OMERO root user by going to http://localhost/
in your web browser.

In addition some example init.d scripts are available should you wish to start OMERO and OMERO.web automatically:

```bash
cp omero-init.d /etc/init.d/omero
chmod a+x /etc/init.d/omero

cp omero-web-init.d /etc/init.d/omero-web
chmod a+x /etc/init.d/omero-web

update-rc.d -f omero remove
update-rc.d -f omero defaults 98 02
update-rc.d -f omero-web remove
update-rc.d -f omero-web defaults 98 02
```

omero-init.d
omero-web-init.d

**Securing OMERO**

The following steps are run as root.

If multiple users have access to the machine running OMERO you should restrict access to OMERO.server’s configuration and runtime directories, and optionally the OMERO data directory:

```bash
chmod go-rwx ~omero/OMERO.server/etc ~omero/OMERO.server/var

# Optionally restrict access to the OMERO data directory
#chmod go-rwx "$OMERO_DATA_DIR"
```

**Regular tasks**

The following steps are run as root.

The default OMERO.web session handler uses temporary files to store sessions which should be deleted at regular intervals, for instance by creating a cron job:

```bash
OMERO_USER=omero
OMERO_SERVER=/home/omero/OMERO.server
su - ${OMERO_USER} -c "${OMERO_SERVER}/bin/omero web cleArsessions"
```

Copy the following commands into the appropriate location:

```bash
cp omero-web-cron /etc/cron.daily/omero-web
chmod a+x /etc/cron.daily/omero-web
```
6.1.8 OMERO.server installation on OS X with Homebrew

Overview

This walk-through demonstrates how to install OMERO on a clean Mac OS X system (10.9 or later) using Homebrew. Note that this demonstrates how to install OMERO.server from the source code via Homebrew, in addition to all its prerequisites.

These instructions are implemented in a series of automated scripts\(^{39}\) which install OMERO via Homebrew from a fresh configuration.

Prerequisites

Xcode

Homebrew requires the latest version of Xcode. Install Xcode and the Command Line Tools for Xcode from the App Store. If you have already installed it, make sure all the latest updates are installed.

Java

Oracle Java may be downloaded from the Oracle website\(^ {40} \).

After installing JDK 7 or JDK 8, check your installation works by running:

```bash
$ java -version
java version "1.8.0_31"
Java(TM) SE Runtime Environment (build 1.8.0_31-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.31-b07, mixed mode)

$ javac -version
javac 1.8.0_31
```

Requirements

All the requirements for OMERO will be installed under `/usr/local`. See also: Installation instructions on the Homebrew wiki\(^ {41} \).

Install Homebrew and make sure `/usr/local/bin` is prepended to your PATH:

```bash
$ ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
$ export PATH=/usr/local/bin:$PATH
```

Update Homebrew and run ‘brew doctor’ to fix potential issues beforehand:

\(^ {39}\)https://github.com/ome/omero-install/tree/develop/osx
\(^ {40}\)http://www.oracle.com/technetwork/java/javase/downloads/index.html
\(^ {41}\)https://github.com/Homebrew/homebrew/blob/master/share/doc/homebrew/Installation.md
$ brew update
$ brew doctor

Install git if not already present:

$ brew list | grep ".git" || brew install git

Install PostgreSQL database server:

$ export LANG=${LANG:-en_US.UTF-8}
$ export LANGUAGE=${LANGUAGE:-en_US:en}
$ brew install postgresql

You should install OMERO using Python 2.7 provided by Homebrew since it makes using Homebrew-provided modules simpler, for example the Ice python bindings needed by OMERO. For a more thorough description of the Homebrew solution, see the Homebrew and Python page. Note that the automated script linked above tests the OMERO installation using the Homebrew Python.

To install the Python provided by Homebrew:

$ brew install python

Check that Python is working and is version 2.7:

$ which python
/usr/local/bin/python
$ python --version
Python 2.7.9

The installation of OMERO via Homebrew depends upon two alternate repositories containing extra formulae: https://github.com/Homebrew/homebrew-science for the HDF5 formula and https://github.com/ome/homebrew-alt for all the OME-provided formulae and older versions of Ice. To add these, run:

$ brew tap homebrew/science
$ brew tap ome/alt

Note: The Homebrew formulae used below provide Python bindings. As described in Homebrew and Python, you should not be in an active virtual environment when you brew install them.

See the step01_deps.sh script for the steps described above.

**OMERO installation**

**OMERO 5.2.6 server**

To install and deploy the 5.2.6 release of OMERO.server, run:

$ brew install omero52 --with-nginx --with-cpp
$ export PYTHONPATH=$(brew --prefix omero52)/lib/python
$ export ICE_CONFIG=$(brew --prefix omero52)/etc/ice.config

---

42 https://github.com/Homebrew/homebrew/blob/master/share/doc/homebrew/Homebrew-and-Python.md
43 https://github.com/Homebrew/homebrew/blob/master/share/doc/homebrew/Homebrew-and-Python.md
This will install the OMERO server to /usr/local/Cellar/omero, which means you will find the log files in /usr/local/Cellar/omero/[release]/var/log. The binaries will be linked to /usr/local/bin:

```
$ which omero
/usr/local/bin/omero
```

Install OMERO python dependencies:

```
$ pip install -r $(brew --prefix omero52)/share/web/requirements-py27-nginx.txt
$ cd /usr/local
$ bash bin/omero_python_deps
```

Start database server:

```
$ pg_ctl -D /usr/local/var/postgres -l /usr/local/var/postgres/server.log -w start
```

Create database and user:

```
$ createuser -w -D -R -S db_user
$ createdb -E UTF8 -O db_user omero_database
$ psql -h localhost -U db_user -l
```

Set database parameters in OMERO:

```
$ omero config set omero.db.name omero_database
$ omero config set omero.db.user db_user
$ omero config set omero.db.pass db_password
```

Create and run script to initialize the OMERO database:

```
$ export ROOT_PASSWORD=${ROOT_PASSWORD:-omero}
$ export PSQL_SCRIPT_NAME=${PSQL_SCRIPT_NAME:-OMERO.sql}
$ omero db script --password $ROOT_PASSWORD -f $PSQL_SCRIPT_NAME
$ psql -h localhost -U db_user omero_database < $PSQL_SCRIPT_NAME
$ rm $PSQL_SCRIPT_NAME
```

Set up OMERO data directory:

```
$ export OMERO_DATA_DIR=${OMERO_DATA_DIR:-~/OMERO.data}
$ mkdir -p $OMERO_DATA_DIR
$ omero config set omero.data.dir $OMERO_DATA_DIR
```

See the OMERO installation script `step02_omero.sh`

**Development server**

If you wish to build OMERO.server from source for development purposes, using the git repository, first use Homebrew to install the OMERO dependencies:

```
$ brew install --only-dependencies omero
```

The default version of Ice installed by the OMERO formula is currently Ice 3.5.

Prepare a place for your OMERO code to live, e.g.:
$ mkdir -p ~/code/projects
$ cd ~/code/projects

If you installed Ice 3.5, you will need to set SLICEPATH to be able to build the server, i.e. export SLICEPATH=/usr/local/share/Ice-3.5/slice.

If you want the development version of OMERO.server, you can clone the source code from the project’s GitHub account to build locally:

$ git clone --recursive git://github.com/openmicroscopy/openmicroscopy
$ cd openmicroscopy && ./build.py

Note: If you have a GitHub account and you plan to develop code for OMERO, you should make a fork into your own account and then clone this fork to your local development machine, e.g.

$ git remote add git://github.com/YOURNAMEHERE/openmicroscopy
$ cd openmicroscopy && ./build.py

See also:
Installing OMERO from source Developer documentation page on how to check out to source code
Build System Developer documentation page on how to build the OMERO.server

Then prepend the development bin directory to your PATH to pick the right executable:

$ export PATH=~/code/projects/openmicroscopy/dist/bin:$PATH

and follow the steps for setting up the database and OMERO data directory as mentioned in the previous section.

OMERO.web

Basic setup for OMERO using Nginx:

$ export HTTPPORT=${HTTPPORT:-8080}
$ omero web config nginx-development --http $HTTPPORT > $(brew --prefix omero52)/etc/nginx.conf

See installation script step03_nginx.sh

For detailed instructions on how to deploy OMERO.web in a production environment such as Apache or Nginx please see OMERO.web deployment.

Note: The internal Django webserver can be used for evaluation and development. In this case please follow the instructions under OMERO.web deployment for developers.

Startup/Shutdown

If necessary start PostgreSQL database server:

$ pg_ctl -D /usr/local/var/postgres -l /usr/local/var/postgres/server.log -w start

Start OMERO:
Start OMERO.web:

$ omero web start
$ nginx -c $(brew --prefix omero52)/etc/nginx.conf

Now connect to your OMERO.server using OMERO.insight or OMERO.web with the following credentials:

U: root
P: omero

Stop OMERO.web:

$ nginx -c $(brew --prefix omero52)/etc/nginx.conf -s stop
$ omero web stop

Stop OMERO:

$ omero admin stop

See example script for a basic functionality test: step04_test.sh

Common issues

General considerations

If you run into problems with Homebrew, you can always run:

$ brew update
$ brew doctor

Also, please check the Homebrew Bug Fixing Checklist\(^4\).

Below is a non-exhaustive list of errors/warnings specific to the OMERO installation. Some if not all of them could possibly be avoided by removing any previous OMERO installation artifacts from your system.

Database

Check to make sure the database has been created and ‘UTF8’ encoding is used

$ psql -h localhost -U db_user -l

This command should give similar output to the following:

<table>
<thead>
<tr>
<th>Name</th>
<th>Owner</th>
<th>Encoding</th>
<th>Collation</th>
<th>Ctype</th>
<th>Access privileges</th>
</tr>
</thead>
<tbody>
<tr>
<td>omero_database</td>
<td>db_user</td>
<td>UTF8</td>
<td>en_GB.UTF-8</td>
<td>en_GB.UTF-8</td>
<td></td>
</tr>
</tbody>
</table>

\(^4\)https://github.com/mxcl/homebrew/wiki/Bug-Fixing-Checklist

6.1. Server installation on UNIX (and UNIX-like platforms)
Macports/Fink

Warning: It appears you have MacPorts or Fink installed.

Follow uninstall instructions from the Macports guide⁴⁵.

PostgreSQL

If you encounter this error during installation of PostgreSQL:

Error: You must `brew link ossp-uuid` before postgresql can be installed

try:

$ brew cleanup
$ brew link ossp-uuid

szip

If you encounter an MD5 mismatch error similar to this:

```bash
==> Installing hdf5 dependency: szip
Already downloaded: /Library/Caches/Homebrew/szip-2.1.tar.gz
Error: MD5 mismatch
Expected: 902f831bcefb69c6b635374424acbead
Got: 0d6a55bb7787f988b9d608f23ef5be0
Archive: /Library/Caches/Homebrew/szip-2.1.tar.gz
(To retry an incomplete download, remove the file above.)
```

then manually remove the archived version located under /Library/Caches/Homebrew, since the maintainer may have updated the file.

numexpr (and other Python packages)

If you encounter an issue related to numexpr complaining about NumPy having too low a version number, verify that you have not previously installed any Python packages using `pip`. In the case where `pip` has been installed before Homebrew, uninstall it:

$ sudo pip uninstall pip

⁴⁵http://guide.macports.org/chunked/installing.macports.uninstalling.html
and then try running `python_deps.sh` again. That should install `pip` via Homebrew and put the Python packages in correct locations.

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 6.1.9 OMERO.server binary repository

#### About

The OMERO.server binary data repository is a fundamental piece of server-side functionality. It provides optimized and indexed storage of original file, pixel and thumbnail data, attachments and full-text indexes. The repository’s directories contain various files that, together with your SQL database, constitute the information about your users and their data that OMERO.server relies upon for normal operation.

#### Layout

The repository is internally laid out as follows:

```
/OMERO
  /OMERO/Pixels    --- Pixel data and pyramids
  /OMERO/Files     --- Original file data
  /OMERO/Thumbnails --- Thumbnail data
  /OMERO/FullText   --- Lucene full text search index
  /OMERO/ManagedRepository --- OMERO.fs filesets, with import logs
  /OMERO/BioFormatsCache --- Cached Bio-Formats state for rendering
```

**Your repository is not:**

- the “database”
- the directory where your OMERO.server binaries are
- the directory where your OMERO.client (OMERO.insight or OMERO.importer) binaries are
- your PostgreSQL data directory

#### PixelService resolution order for locating binary data for images

When the server is trying to find the binary data for an image, it looks:

- first under `/OMERO/Pixels` for a `$NUMBER_pyramid` file
- then under `/OMERO/Pixels` for a regular `$NUMBER` file
- then under `/OMERO/Files` for OMERO 4 files
- or under `/OMERO/ManagedRepository` for OMERO 5 files

#### Locking and remote shares

The OMERO server requires proper locking semantics on all files in the binary repository. In practice, this means that remotely mounted shares such as AFS, CIFS, and NFS can cause issues. If you have experience and/or the time to manage and monitor the locking implementations of your remote filesystem, then using them as for your binary repository should be fine.

If, however, you are seeing errors such as `NullPointerExceptions`, “Bad file descriptors” and similar in your server log, then you will need to use directly connected disks.
**Warning:** If your binary repository is a remote share and mounting the share fails or is dismounted, OMERO will continue operating using the mount point instead! To prevent this, make the mount point read-only for the OMERO user so that no data can be written to the mount point.

---

**Changing your repository location**

**Note:** It is strongly recommended that you make all changes to your OMERO binary repository with the server shut down. Changing the `omero.data.dir` configuration does not move the repository for you, you must do this yourself.

Your repository location can be changed from its `/OMERO` default by modifying your OMERO.server configuration as follows:

```
$ cd OMERO.server
$ bin/omero config set omero.data.dir /mnt/really_big_disk/OMERO
```

The suggested procedure is to shut down your OMERO.server instance, move your repository, change your `omero.data.dir` and then start the instance back up. For example:

```
$ cd OMERO.server
$ bin/omero admin stop
$ mv /OMERO /mnt/really_big_disk
$ bin/omero config set omero.data.dir /mnt/really_big_disk/OMERO
$ bin/omero admin start
```

The `omero.managed.dir` property for the OMERO.fs managed repository may be adjusted similarly, even to a directory outside `omero.data.dir`.

**Note:** The managed repository should be located and configured to allow the OMERO server processes fast access to the uploaded filesets that it contains.

---

**Access permissions**

Your repository should be owned by the same user that is starting your OMERO.server instance. This is often either yourself (find out by executing `whoami`) or a separate `omero` (or similar) user who is dedicated to running OMERO.server. For example:

```
$ whoami
omero
$ ls -al /OMERO
total 24
drwxr-xr-x 5 omero omero 128 Dec 12 2006 .
```

**Repository size**

At minimum, the binary repository should be comfortably larger than the images and other files that users may be uploading to it. It is fine to set `omero.data.dir` or `omero.managed.dir` to very large volumes, or to use logical volume management to conveniently increase space as necessary.

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.
6.1.10 OMERO.server and PostgreSQL

In order to be installed, OMERO.server requires a running PostgreSQL instance that is configured to accept connections over TCP. This section explains how to ensure that you have the correct PostgreSQL version and that it is installed and configured correctly.

Ensuring you have a valid PostgreSQL version

For OMERO 5.2, PostgreSQL version 9.4 or later is recommended. Make sure you are using a supported version. You can check which version of PostgreSQL you have installed with any of the following commands:

```
$ createuser -V
createuser (PostgreSQL) 9.4.1
$ psql -V
psql (PostgreSQL) 9.4.1
$ createdb -V
createdb (PostgreSQL) 9.4.1
```

If your existing PostgreSQL installation is an earlier version, it is recommended that you upgrade to a more up-to-date version. Before upgrading, stop the OMERO server and then perform a full dump of the database using `pg_dump`. See the OMERO.server backup and restore section for further details.

If using a Linux distribution-provided PostgreSQL server, upgrading to a newerversion of the distribution will usually make a newerversion of PostgreSQL available. If the database was not migrated to the new version automatically, restore your backup after installing, configuring and starting the new version of the database server. If a PostgreSQL server was not provided by your system, EnterpriseDB provide an installer.

Checking PostgreSQL port listening status

You can check if PostgreSQL is listening on the default port (TCP/5432) by running the following command:

```
$ netstat -an | egrep '5432.*LISTEN'
tcp 0 0 0.0.0.0:5432 0.0.0.0:* LISTEN
tcp 0 0 :::5432 :::* LISTEN
```

**Note:** The exact output of this command will vary. The important thing to recognize is whether or not a process is listening on TCP/5432.

If you cannot find a process listening on TCP/5432 you will need to find your `postgresql.conf` file and enable PostgreSQL's TCP listening mode. The exact location of the `postgresql.conf` file varies between installations.

It may be helpful to locate it using the package manager (rpm or dpkg) or by utilizing the `find` command. Usually, the PostgreSQL data directory (which houses the `postgresql.conf` file, is located under `/var` or `/usr`:

```
$ sudo find /etc -name 'postgresql.conf'
$ sudo find /usr -name 'postgresql.conf'
$ sudo find /var -name 'postgresql.conf'
/var/lib/postgresql/data/postgresql.conf
```

**Note:** The PostgreSQL data directory is usually only readable by the user `postgres` so you will likely have to be `root` in order to find it.

Once you have found the location of the `postgresql.conf` file on your particular installation, you will need to enable TCP listening. The area of the configuration file you are concerned about should look similar to this:

```text
# A TCP connection is configured on the `listen` setting.
listen = 'tcp://0.0.0.0:5432'
```

---

46 [http://www.postgresql.org/support/versioning/](http://www.postgresql.org/support/versioning/)
#listen_addresses = 'localhost'  # what IP address(es) to listen on;
# comma-separated list of addresses;
# defaults to 'localhost', '*' = all

#port = 5432

max_connections = 100

# note: increasing max_connections costs ~400 bytes of shared memory per
# connection slot, plus lock space (see max_locks_per_transaction). You
# might also need to raise shared_buffers to support more connections.
#superuser_reserved_connections = 2
#unix_socket_directory = *
#unix_socket_group = *
#unix_socket_permissions = 0777 # octal
#bonjour_name = * # defaults to the computer name

PostgreSQL HBA (host based authentication)

OMERO server must have permission to connect to the database that has been created in your PostgreSQL instance. This
is configured in the host based authentication file, pg_hba.conf. Check the configuration by examining the contents of
pg_hba.conf. It’s important that at least one line allows connections from the loopback address (127.0.0.1) as follows:

```
# TYPE DATABASE USER CIDR-ADDRESS METHOD
# IPv4 local connections:
host all all 127.0.0.1/32 md5
# IPv6 local connections:
host all all ::1/128 md5
```

Note: The other lines that are in your pg_hba.conf are important either for PostgreSQL internal commands to work or for
existing applications you may have. Do not delete them.

Completing configuration

After making any configuration changes to postgresql.conf or pg_hba.conf, reload the server for the changes to take
effect.

`$ sudo service postgresql reload`

See also:

PostgreSQL

Connections and Authentication

Client Authentication

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event
of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.1.11 OMERO.web deployment

OMERO.web is the web application component of the OMERO platform which allows for the management, visualization (in a
fully multi-dimensional image viewer) and annotation of images from a web browser. It also includes the ability to manage users
and groups.

OMERO.web is an integral part of the OMERO platform and can be deployed with:
• WSGI\textsuperscript{51} using a WSGI capable web server such as nginx\textsuperscript{52} and gunicorn\textsuperscript{53} (you can also use Apache 2.2\textsuperscript{54} with mod_wsgi\textsuperscript{55} enabled but this is not recommended for new installations as support has been deprecated as of OMERO 5.2.6 and is likely to be dropped during the 5.3.x line)
• The built-in Django lightweight development server (for testing only; see the OMERO.web deployment for developers page).

If you need help configuring your firewall rules, see the Server security and firewalls page.

Prerequisites

• OMERO and its prerequisites (see OMERO.server installation).
• Python 2.7
  – Pillow\textsuperscript{56} should be available for your distribution. We currently do not support version 3.0+.
  – Matplotlib\textsuperscript{57} should be available for your distribution
• A WSGI capable web server

Deployment (Unix/Linux)

There are a number of good ways to easily deploy OMERO.web. If you have installed Nginx or Apache, OMERO can automatically generate a configuration file for your web server. The location of the file will depend on your system, please refer to your web server’s manual. See Customizing your OMERO.web installation for additional customization options.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

DEPRECATED: OMERO.web Apache and mod_wsgi deployment (Unix/Linux)

Apache 2.2+ with mod_wsgi configuration (Unix/Linux)

Note: Since OMERO 5.2, the OMERO.web framework no longer bundles a copy of the Django package, instead manual installation of the Django dependency is required. It is highly recommended to use Django 1.8\textsuperscript{58} (LTS) which requires Python 2.7. For more information see Python on the Version requirements page.

Also note that support for Apache deployment is deprecated as of OMERO 5.2.6 and is likely be dropped during the 5.3.x line.

OMERO.web uses the Web Server Gateway Interface (WSGI)\textsuperscript{59} as a primary deployment platform. The Python standard PEP 3333\textsuperscript{60} that describes how a web server communicates with web applications.

Apache default configuration (Unix/Linux) Install Django 1.8\textsuperscript{61} using package requirements file:

$ pip install -r share/web/requirements-py27-apache.txt

Note: For more details refer to how to install Django 1.8\textsuperscript{62} or how to upgrade Django to 1.8\textsuperscript{63}.

Additional settings can be configured by changing the following properties:

\[\text{http://wsgi.readthedocs.org}\]
\[\text{http://nginx.org/}\]
\[\text{http://docs.gunicorn.org/}\]
\[\text{http://httpd.apache.org/}\]
\[\text{http://www.modwsgi.org/}\]
\[\text{http://matplotlib.org/}\]
\[\text{https://docs.djangoproject.com/en/1.8/releases/1.8/}\]
\[\text{http://wsgi.readthedocs.org/en/latest/learn.html}\]
\[\text{https://www.python.org/dev/peps/pep-3333/}\]
\[\text{https://docs.djangoproject.com/en/1.8/releases/1.8/}\]
\[\text{https://docs.djangoproject.com/en/1.8/topics/install/#install-the-django-code}\]
\[\text{https://docs.djangoproject.com/en/1.8/topics/install/#remove-any-old-versions-of-django}\]
OMERO Documentation, Release 5.2.6

- `omero.web.application_server.max_requests` to 500
- `omero.web.wsgi_workers` to `(2 \times \text{NUM\_CORES}) + 1`

**Note:** Do not scale the number of workers to the number of clients you expect to have. OMERO.web should only need 4-12 worker processes to handle many requests per second.

### Apache configuration (Unix/Linux)

If you have installed Apache, install mod_wsgi\(^{64}\).

OMERO can automatically generate a configuration file for your web server. The location of the file will depend on your system, please refer to your web server’s manual. See **Customizing your OMERO.web installation** for additional customization options.

Set the following:

```
$bin/omero config set omero.web.application_server "wsgi"
```

Creates symlinks for static media files

```
$bin/omero web syncmedia
```

To create a site configuration file for inclusion in the main Apache configuration redirect the output of the following command into a file:

For Apache 2.4, run the following command:

```
$ bin/omero web config apache24 > httpd-omero.conf
```

For Apache 2.2, run the following command:

```
$ bin/omero web config apache > httpd-omero.conf
```

**Configuration file:**

```
<VirtualHost _default_:80>

WSGIDaemonProcess omeroweb processes=5 threads=1 maximum-requests=0 display-name=%{GROUP} user=omero ... Ice:/home/omero/OMERO.server/lib/python:/home/omero/OMERO.server/lib/fallback:/home/omero/OMERO.server/lib/python/omeroweb

WSGIScriptAlias / /home/omero/OMERO.server/lib/python/omeroweb/wsgi.py process-group=omeroweb

<Directory "/home/omero/OMERO.server/lib/python/omeroweb">

WSGIProcessGroup omeroweb
WSGIApplicationGroup %{GLOBAL}
Order allow,deny
Allow from all
</Directory>

Alias /static /home/omero/OMERO.server/lib/python/omeroweb/static
<Directory "/home/omero/OMERO.server/lib/python/omeroweb/static">

Options -Indexes FollowSymLinks
Order allow,deny
Allow from all
</Directory>

</VirtualHost>
```

# see https://code.google.com/p/modwsgi/wiki/ConfigurationIssues
WSGISocketPrefix run/wsgi

---

\(^{64}\)http://www.modwsgi.org/

# Use this on Ubuntu/Debian/MacOSX systems:
# WSGISocketPrefix /var/run/wsgi

To configure an HTTPS server follow the Apache documentation\(^{65}\).
Then reload Apache.

**Running OMERO.web (Unix/Linux)**  OMERO.web is used in ‘daemon’ mode where UNIX sockets are used to communicate between the Apache child processes and the daemon processes which are to handle a request. **OMERO.web does not provide any management for these ‘daemon’ processes; bin/omero web start/status/stop will not work as they are for managing gunicorn processes only.**

**Note:** **This documentation is for OMERO 5.2.** This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### OMERO.web Nginx and Gunicorn deployment (Unix/Linux)

**Note:** Since OMERO 5.2, the OMERO web framework no longer bundles a copy of the Django package, instead manual installation of the Django dependency is required. It is highly recommended to use Django 1.8\(^{66}\) (LTS) which requires Python 2.7. For more information see Python on the Version requirements page.

OMERO.web uses the Web Server Gateway Interface (WSGI)\(^{67}\) as a primary deployment platform. The Python standard PEP 3333\(^{68}\) that describes how a web server communicates with web applications.

**Gunicorn default configuration (Unix/Linux)** Install Django 1.8\(^{69}\) and Gunicorn >= 19.3\(^{70}\) using the package requirements file:

```
$ pip install -r share/web/requirements-py27-nginx.txt
```

**Note:** For more details refer to how to install Django 1.8\(^{71}\) or upgrade Django to 1.8\(^{72}\).

Additional settings can be configured by changing the following properties:

- `omero.web.application_server.max_requests` to 500
- `omero.web.wsgi_workers` to `(2 x NUM_CORES) + 1`

**Note:** Do not scale the number of workers to the number of clients you expect to have. OMERO.web should only need 4-12 worker processes to handle many requests per second.

- `omero.web.wsgi_args` Additional arguments. For more details check Gunicorn Documentation\(^{73}\).

### Nginx configuration (Unix/Linux)  If you have installed Nginx, OMERO can automatically generate a configuration file for your web server. The location of the file will depend on your system, please refer to your web server’s manual. See Customizing your OMERO.web installation for additional customization options.

\(^{65}\)http://httpd.apache.org/docs/trunk/mod/mod_ssl.html
\(^{66}\)https://docs.djangoproject.com/en/1.8/releases/1.8/
\(^{67}\)http://wsgi.readthedocs.org/en/latest/learn.html
\(^{68}\)https://www.python.org/dev/peps/pep-3333/
\(^{69}\)https://docs.djangoproject.com/en/1.8/releases/1.8/
\(^{70}\)http://gunicorn.org
\(^{71}\)https://docs.djangoproject.com/en/1.8/topics/install/#install-the-django-code
\(^{72}\)https://docs.djangoproject.com/en/1.8/topics/install/#remove-any-old-versions-of-django
To create a site configuration file for inclusion in a system-wide nginx configuration redirect the output of the following command into a file:

```
$ bin/omero web config nginx
```

```
upstream omeroweb {
    server 127.0.0.1:4080 fail_timeout=0;
}

server {
    listen 80;
    server_name $hostname;
    sendfile on;
    client_max_body_size 0;

    # maintenance page serve from here
    location @maintenance {
        root /home/omero/OMERO.server/etc/templates/error;
        try_files $uri /maintainance.html =502;
    }

    # weblitz django apps serve media from here
    location /static {
        alias /home/omero/OMERO.server/lib/python/omeroweb/static;
    }

    location @proxy_to_app {
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_redirect off;
        proxy_buffering off;

        proxy_pass http://omeroweb;
    }

    location / {

        error_page 502 @maintenance;
        # checks for static file, if not found proxy to app
        try_files $uri @proxy_to_app;
    }
}
```

**Note:** OMERO.web requires body_in_file_only adjusted in your default nginx config because nginx must buffer incoming data. Make sure you have that set to the following config:

```
http {
    ...
    sendfile on;
    send_timeout 60s;
    client_max_body_size 0;
    ...
}
```

To configure an HTTPS server follow the nginx documentation\(^4\).

Running OMERO.web (Unix/Linux)   Start the Gunicorn worker processes listening by default on 127.0.0.1:4080:

$ bin/omero web start
... static files copied to '/home/omero/OMERO.server/lib/python/omeroweb/static'.
Starting OMERO.web... [OK]

The Gunicorn workers are managed separately from other OMERO.server processes. You can check their status or stop them using the following commands:

$ bin/omero web status
OMERO.web status... [RUNNING] (PID 59217)
$ bin/omero web stop
Stopping OMERO.web... [OK]
Django WSGI workers (PID 59217) killed.

Download limitations   In order to offer users the ability to download data from OMERO.web you have to deploy using EXPERIMENTAL: Async workers. OMERO.web is able to handle multiple clients on a single worker thread switching context as necessary while streaming binary data from OMERO.server. Depending on the traffic and scale of the repository you should configure connections and speed limits on your server to avoid blocking resources. We recommend you run benchmark and performance tests. It is also possible to apply Download restrictions and offer alternative access to binary data.

Note: Handling streaming request/responses requires proxy buffering to be turned off. For more details refer to Gunicorn deployment and Nginx configuration.

Note: omero.web.application_server.max_requests should be set to 0

Benchmark   We run example benchmarks on rendering thumbnails and 512x512 pixels planes for 100 concurrent users making 5000 requests in total:

$ ab -n 5000 -c 100 https://server.openmicroscopy.org/omero/webclient/render_thumbnail/size/96/1234/
This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Server Software:   nginx/1.9.9
Server Hostname:   server.openmicroscopy.org
Server Port:   80
Document Path:   /omero/webclient/render_thumbnail/size/96/31851/
Document Length:   1880 bytes
ConcURRENCY Level:   100
Time taken for tests:   224.488 seconds
Complete requests:   5000
Failed requests:   0
Write errors:   0
Total transferred:   10450000 bytes
HTML transferred:   9400000 bytes
Requests per second:   22.27 [#/sec] (mean)
Time per request:   4489.763 [ms] (mean)
Time per request:   44.898 [ms] (mean, across all concurrent requests)
Transfer rate:   45.46 [Kbytes/sec] received

Connection Times (ms)
min  mean[+/-sd] median  max

Connect: 0 0 0.3 0 3
Processing: 435 4446 685.2 4363 7644
Waiting: 432 4446 685.3 4362 7644
Total: 435 4446 685.1 4363 7644

Percentage of the requests served within a certain time (ms)
50% 4363
66% 4553
75% 4670
80% 4750
90% 5072
95% 5398
98% 6795
99% 6955
100% 7644 (longest request)

$ ab -n 5000 -c 100 http://server.openmicroscopy.org/omero/webclient/render_image/1234/20/0/
This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Server Software: nginx/1.9.9
Server Hostname: server.openmicroscopy.org
Server Port: 80

Document Path: /omero/webclient/render_image/1234/20/0/
Document Length: 24293 bytes

Concurrency Level: 100
Time taken for tests: 247.154 seconds
Complete requests: 5000
Failed requests: 0
Write errors: 0
Total transferred: 122515000 bytes
HTML transferred: 121465000 bytes
Requests per second: 20.23 [#/sec] (mean)
Time per request: 4943.080 [ms] (mean)
Time per request: 49.431 [ms] (mean, across all concurrent requests)
Transfer rate: 484.09 [Kbytes/sec] received

Connection Times (ms)
 min mean[+-sd] median max
Connect: 0 0 0.4 0 4
Processing: 482 4898 855.3 4737 8303
Waiting: 476 4898 855.3 4737 8303
Total: 482 4898 855.2 4737 8303

Percentage of the requests served within a certain time (ms)
50% 4737
66% 5041
75% 5250
80% 5397
90% 5862
95% 6621
98% 7301
99% 8062
100% 8303 (longest request)

Troubleshooting  In order to identify why OMERO.web is not available run:
$ bin/omero web status

Then consult nginx error.log and OMERO.server/var/log/OMEROweb.log

For more details check OMERO.web issues.

**Debugging**  To run the WSGI server in debug mode, enable Gunicorn logging

```bash
$ bin/omero config set omero.web.wsgi_args "--log-level=DEBUG --error-logfile=/home/omero/OMERO.server/var/log/error.log".
```

**EXPERIMENTAL: Gunicorn advance configuration (Unix/Linux)**

Note: Experimental configurations are not ready for production use. Configuration may change. Features may not work properly.

OMERO.web deployment can be configured with sync and async workers. Sync workers are faster and recommended for a data repository with Download restrictions. If you wish to offer users the ability to download data then you have to use async workers; read more about Download limitations.

For more details refer to Gunicorn design.

**EXPERIMENTAL: Sync workers**

Note: Experimental configurations are not ready for production use. Configuration may change. Features may not work properly.

- Install futures

```bash
$ pip install futures
```

Additional settings can be configured by changing the following properties:

- The number of worker threads for handling requests, see Gunicorn threads

```bash
$ bin/omero config set omero.web.wsgi_worker_class
$ bin/omero config set omero.web.wsgi_threads $(2-4 x NUM_CORES)
```

**EXPERIMENTAL: Async workers**

Note: Experimental configurations are not ready for production use. Configuration may change. Features may not work properly.

- Install Gevent >= 0.13

```bash
$ pip install "gevent>=0.13"
```

Additional settings can be configured by changing the following properties:

- The maximum number of simultaneous clients, see Gunicorn worker-connections

```bash
$ bin/omero config set omero.web.wsgi_worker_class gevent
$ bin/omero config set omero.web.wsgi_worker_connections 1000
$ bin/omero config set omero.web.application_server.max_requests 0
```

---

79 [https://pypi.python.org/pypi/futures](https://pypi.python.org/pypi/futures)
Loggin in to OMERO.web

Once you have deployed and started the server, you can use your browser to access the OMERO.web client:

- [http://your_host/omero](http://your_host/omero) OR, for development server: [http://localhost:4080](http://localhost:4080)

![OMERO.web login](image)

Figure 6.1: OMERO.web login

**OMERO.web Maintenance (Unix/Linux)**

- **Session cookies** `omero.web.session_expire_at_browser_close`:
  - A boolean that determines whether to expire the session when the user closes their browser. See [Django Browser-length sessions vs. persistent sessions documentation](https://docs.djangoproject.com/en/1.8/topics/http/sessions/#browser-length-vs-persistent-sessions) for more details. Default: True.

  ```
  $ bin/omero config set omero.web.session_expire_at_browser_close "True"
  ```

  - The age of session cookies, in seconds. Default: 86400.

  ```
  $ bin/omero config set omero.web.session_cookie_age 86400
  ```

- **Session engine**:
  - Each session for a logged-in user in OMERO.web is kept in the session store. Stale sessions can cause the store to grow with time. OMERO.web uses by default the OS file system as the session store backend and does not automatically purge stale sessions, see [Django file-based session documentation](https://docs.djangoproject.com/en/1.8/topics/http/sessions/#using-file-based-sessions) for more details. It is therefore the responsibility of the OMERO administrator to purge the session cache using the provided management command:

  ```
  $ bin/omero web clearsessions
  ```

  It is recommended to call this command on a regular basis, for example as a *daily cron job*, see [Django clearing the session store documentation](https://docs.djangoproject.com/en/1.8/topics/http/sessions/#clearing-the-session-store) for more information.

---
OMERO.web offers alternative session backends to automatically delete stale data using the cache session store backend, see Django cached session documentation\(^\text{87}\) for more details. After installing all the cache prerequisites set the following:

\[
\text{\$ bin/omero config set omero.web.session_engine django.contrib.sessions.backends.cache}
\]

* Memcached\(^\text{88}\):

\[
\text{\$ bin/omero config set omero.web.caches "default": "django.core.cache.backends.memcached.MemcachedCache", "LOCATION": "127.0.0.1:11211", "TIMEOUT": "86400"}
\]

* EXPERIMENTAL: Redis 2.4+\(^\text{89}\) requires django-redis-cache 1.6.5+\(^\text{90}\):

\[
\text{\$ pip install django-redis-cache>1.6.5}
\text{\$ bin/omero config set omero.web.caches "default": "redis_cache.RedisCache", "LOCATION": "127.0.0.1:6379"}
\]

**Note:** The generated Nginx and Apache configuration files will automatically display a maintenance page if an attempt is made to access OMERO.web whilst it is not running.

### Customizing your OMERO.web installation

By default OMERO.web expects to be run from the root URL of the web server. This can be changed by setting `omero.web.prefix` and `omero.web.static_url`. For example, to make OMERO.web appear at `http://example.org/omero/`:

\[
\text{\$ bin/omero config set omero.web.prefix '/omero'}
\text{\$ bin/omero config set omero.web.static_url '/omero/static/'}
\]

and regenerate your web-server configuration (see *Deployment (Unix/Linux)*).

The front-end webserver (Nginx, Apache) can be setup to run on a different host from OMERO.web. You will need to set `omero.web.application_server.host` to ensure OMERO.web is accessible on an external IP.

All configuration options can be found on various sections of Web developers documentation. For the full list, refer to *Web properties* or:

\[
\text{\$ bin/omero web -h}
\]

**Note:** Depending on the deployment choice, OMERO.web will not activate configuration changes until either gunicorn is restarted using `bin/omero web restart (nginx)` or the Apache webserver has been reloaded.

The most popular configuration options include:

- Debug mode, see `omero.web.debug`.
- Customizing OMERO clients e.g. to add your own logo to the login page (`omero.web.login_logo`) or use an index page as an alternative landing page for users (`omero.web.index_template`). See *Customize OMERO clients* for further information.
- Enabling a public user see *Public data in the repository*.

\(^{87}\)https://docs.djangoproject.com/en/1.8/topics/http/sessions/#using-cached-sessions
\(^{88}\)https://memcached.org/
\(^{89}\)http://redis.io/
Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

# Python installation requirements for OMERO.
#===========================================
# pip install -r requirements.txt

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

# Python installation requirements for OMERO.
#===========================================
numexpr==1.4.2
Pillow<3.0

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

# Python installation requirements for OMERO.
#===========================================
requirements_centos6_py27.txt
matplotlib
Pillow<3.0

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

# Python installation requirements for OMERO.
#===========================================
numpy.matplotlib
Pillow<3.0
omego==0.3.0

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.2 Server installation on Windows

Warning: OMERO 5.2 is the last major version which will feature Windows support for OMERO.server and OMERO.web deployment. See this blog post for details. If you are installing a new server, we highly recommend you use a different OS (see Version requirements).

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.2.1 OMERO.server installation

Warning: OMERO 5.2 is the last major version which will feature Windows support for OMERO.server and OMERO.web deployment. See this blog post for details. If you are installing a new server, we highly recommend you use a different OS (see Version requirements).

*http://blog.openmicroscopy.org/tech-issues/future-plans/deployment/2016/03/22/windows-support/

Limitations

Installation will require an “Administrator” level account for which you know the password. If you are unsure of what it means to have an “Administrator” level account, or if you are generally having issues with the various users/passwords described in this install guide, please see Which user account and password do I use where?.
• Unless you are clear on the differences, you should also open all consoles as an administrator to prevent file permission issues.

• Significant testing has taken place on Windows Server 2012 and we recommend this version.

• OMERO.movie is not supported on Windows at present.

• A reboot is required after installing the prerequisites.

Note: The default user paths on Windows usually contain spaces so you will need to ensure that the server installation path has no spaces, C:\OMERO.server for example.

Prerequisites

Note: The installation of these prerequisite applications is largely outside the scope of this document. Do not mix 32bit (x86) and 64bit (amd64) packages - install either all 32bit or all 64bit. Check your JRE as well.

The following are necessary:

Java SE Development Kit (JDK)


Ice

Note: OMERO 5.2 supports Ice 3.5. You will need to pick the matching server download for the version of Ice you have installed. If you are using precompiled packages of Ice and Python (recommended) you should use the versions specified below. In particular since OMERO does not support Python 3 it will not work with the ZeroC Ice 3.5 package for Windows. However, Ice 3.5 will work when built against Python 2.7.

You can download Ice 3.5.1 built against Python 2.7 from our Ice downloads page91, unzip the archive and move the extracted directory to C: \.

If you plan to develop for C++, be sure to read the instructions on the OMERO C++ language bindings page.

91http://downloads.openmicroscopy.org/ice/3.5.1/
Python 2

Note: OMERO does not currently support Python 3.

Make sure you install the correct Python 2 version for your Ice version:

- If you have downloaded Ice 3.5.1 from our downloads page, you will need to install Python 2.792.

Python 2.7.9 is strongly recommended. OMERO has been successfully tested with this version and there were problems noticed with later versions.

As these are the “vanilla” python distributions (no extra libraries), you might need to install further dependencies, making sure to download the correct version (32/64-bit) for your Python distribution.

32/64-bit 32-bit or 64-bit Python may be used. You can obtain 64-bit precompiled dependencies from http://www.lfd.uci.edu/~gohlke/pythonlibs.

Python for Windows extensions Python for Windows extensions (PyWin32) is required. The installer is available from the PyWin download page93.

The version you need is: pywin32-XXX.win32-py2.7.exe (or pywin32-XXX.win-amd64-py2.7.exe for 64-bit Python) where XXX should be the latest release number and A and B stand for the Python version, for example pywin32-218.win32-py2.7.exe.

PostgreSQL (9.4 or higher)

PostgreSQL has to be installed and configured to accept TCP connections. PostgreSQL 9.2 and earlier are not supported. 9.4 or later is recommended.

The One click installer can be found on the PostgreSQL Windows download page94.

- You must install PostgreSQL as a service if you want to follow this document.

1. Run the downloaded installer. You may be prompted for permission to continue with a “user account control” dialog. Click yes to continue. The installer will now start.
2. Choose the installation directory. The default is fine.
3. Choose the data directory. The default is fine, but if you want to keep the data in a specific location, you may choose an alternative location here.
4. Enter a password for the special “postgres” system account. OMERO does not use this account, but you will need to record the password for creating the database, below.
5. Enter the port number for PostgreSQL to listen on for incoming connections. The default, 5432, is fine and should not be changed.
6. Select the locale. The default here is fine.
7. PostgreSQL will now be installed and started.

Environment variables

For the prerequisite software to run properly, your PATH and PYTHONPATH environment variables must be configured. This is particularly important for Ice, which does not set the required variables by default.

Update your Windows environment variables: (REQUIRES RESTART!)

1. Locate the System control panel page on the Start Menu under Settings → Control Panel, open it and navigate to the Advanced tab (on Windows Vista the dialog will be visible after clicking the Change settings link on the System control panel page):

92https://www.python.org/downloads/release/python-279/
93http://sourceforge.net/projects/pywin32/files/pywin32/
94http://www.postgresql.org/download/windows
2. Open the *Environment Variables* dialog by clicking on the *Environment Variables...* button of the above dialog:

3. Edit the existing *System* environment variable *Path* and add a new variable pointing to the Ice installation *bin* directory. At the front of the *Path* variable also add a new string pointing to the Python installation directory (e.g. *C:\Python27*). Then add a brand new *System* environment variable called *PYTHONPATH* pointing to the Ice installation *python* location:

When setting the ENV variables, make sure you write in the correct paths. You must have entries for:

- python (the first PATH entry, e.g. “C:\Python27;%Sys…”)
- icebin (the last PATH entry, e.g. “…;C:\Ice-3.5.1\bin;”)
- PYTHONPATH pointing to the python folder in the ICE installation (e.g. “C:\Ice-3.5.1\python;”)

---

**Figure 6.2: Advanced System Properties**

**Figure 6.3: Environment Variables**
Warning: Remember that the Windows path separator is the semicolon ; and must be appended after every entry. Make sure the first inserted ENV PATH entry (the python path) finishes with a semicolon (eg. “C:\Python27;%SystemRoot%...”) otherwise you could corrupt other system applications.

4. **Restart your computer.** For environment changes to take effect in background services, a restart is unfortunately necessary. See [http://support.microsoft.com/kb/821761](http://support.microsoft.com/kb/821761) for more information.

**Creating a database for OMERO**

- Open pgAdmin III; you can find it on the Start Menu under *Programs → PostgreSQL → pgAdmin III*.
- Connect to the database by double-clicking on the *PostgreSQL* database (or right-click and choose *Connect*) and providing your *postgres* user login password set during the installation, above.
- Create a new (non-superuser) database user for OMERO:
  1. Right-click on *Login Roles* and select *New Login Role* ...
  2. On the *Properties* tab, enter a name for the OMERO database user (referenced as db_user below) as the Role name, for example *omero*. On the *Definition* tab, enter the password for the user twice and then press OK. Make sure you record this username and password for later use; you will need to configure OMERO to use your username and password by setting the `omero.db.name` and `omero.db.pass` properties.
- Create a new database for OMERO:
  1. Right-click on *Databases* and select *New Database ...*
  2. On the *Properties* tab, enter a name for the OMERO database (referenced as omero_database below) as the Name, for example *omero*, then press OK.
  3. Switch to the *Definition* tab and make sure UTF8 is set for the *Encoding* property.

**Location for your OMERO binary repository**

See also:

**OMERO.server binary repository**

- Create a directory for the OMERO binary data repository (*C:\OMERO* is the default location and should be used unless you explicitly have a reason not to and know what you are doing).
- This is *not* where you want the OMERO application to be installed, it is a *separate* directory that OMERO.server will use to store binary data:

  `C:\mkdir OMERO`

- If necessary change the ownership of the directory. *C:\OMERO must* either be owned by the user starting the server or that user must have permission to write to the directory. Please see [OMERO.server binary repository](#) for more details.

When performing some operations the clients make use of temporary file storage and log directories. By default these files are stored below the user’s home directory (on Windows *C:\Users\<username>* in *omero\tmp, omero\log* and *omero\sessions*).

**Note:** If your home directory is stored on a network (NFS mounted or similar), then file read and write operations occur over the network. This can slow access down. Installing OMERO on a network mapped drive is strongly discouraged.

The OMERO.server also accesses the *omero\tmp* and *omero\log* folders of the user account running the server process. As the server makes heavier use of these folders than the clients, if that user’s home folder is stored on a network the server can be slowed down. To get around this for the OMERO.server you can define an *OMERO_TMPDIR* environment variable pointing to a temporary directory located on the local file system e.g. *C:\tmp*.
Installation

• Download and extract the OMERO.server zip file, and note its location. Below it is referred to as C:\OMERO.server.

• Optionally, review Configuration properties glossary, which contains all default settings. Any configuration settings you would like to change can be changed in the next step.

• Change any settings that are necessary using bin\omero config, including the name and/or password for the ‘db_user’ database user you chose above or the database name if it is not “omero_database”. (Quotes are only necessary if the value could be misinterpreted by the shell. See link95).

C:\> cd C:\OMERO.server
C:\OMERO.server> bin\omero config set omero.db.name omero_database
C:\OMERO.server> bin\omero config set omero.db.user db_user
C:\OMERO.server> bin\omero config set omero.db.pass db_password

• If you have chosen a non-standard OMERO binary repository location above, be sure to configure the omero.data.dir property.

When using C:\ style file paths it is necessary to “escape” the backslashes. For example:

C:\> bin\omero config set omero.data.dir C:\\OMERO

• Create the OMERO database initialization script. You will be asked for the version of the script which you would like to generate, where both defaults can be accepted. Finally, you will be asked to enter and confirm password for your newly created OMERO root user.

Warning: For illustrative purposes, the default password for the OMERO rootuser is root_password. However, you should not use this default value for your installation but use your own choice of password instead. This should not be the same password as your Linux/Mac/Windows root user!

C:\> cd C:\OMERO.server
C:\OMERO.server> bin\omero db script --password root_password

Using OMERO5.2 for version
Using 0 for patch
Using password from commandline

Saving to C:\OMERO.server\OMERO5.2_0.sql

The generated SQL file is found in the folder where you called the “omero db script” command. This could cause a permission denied error in the populate db step if the postgres user cannot access that location. Move the file to a different location or use the -f option.

• Initialize your database with the script.

1. Launch SQL Shell (psql) from the Start Menu under Programs → PostgreSQL → SQL Shell (psql)

   Server [localhost]:
   Database [postgres]: omero_database
   Port [5432]:
   Username [postgres]: db_user
   Password for user db_user:
   Welcome to psql 9.3.4, the PostgreSQL interactive terminal.

   Type: copyright for distribution terms
   h for help with SQL commands
   ? for help with psql commands
   g or terminate with semicolon to execute query
   q to quit

95http://www.openmicroscopy.org/community/viewtopic.php?f=5&t=3608&p922

6.2. Server installation on Windows
Warning: Console code page (437) differs from Windows code page (1252) 8-bit characters might not work correctly. See psql reference page ‘Notes for Windows users’ for details.

2. Execute the following to populate your database (the forward slashes are intentional - if you get a permission denied error it is because the path is wrong, not the slashes):

   omero=> SET client_encoding = 'UTF8';
   omero=> \i C:/OMERO.server/OMERO5.2__0.sql
   ...
   ...
   omero=> \q

   • Before starting the OMERO.server, you should run the OMERO diagnostics script so that you check that all of the settings are correct, e.g.

   C:\OMERO.server\> bin\omero admin diagnostics

   The diagnostic tool may say that psql is not found. This should not be a problem but you can fix it by adding its bin folder to the path. For example, C:\Program Files (x86)\PostgreSQL\9.3\bin. Remember to reboot after changing the environment.

   • You can now start the server using:

   C:\OMERO.server> bin\omero admin start
   Creating C:\OMERO.server\var\master
   Creating C:\OMERO.server\var\registry
   No descriptor given. Using etc\grid\windefault.xml
   Installing OMERO.master Windows service.
   Successfully installed OMERO.master Windows service.
   Starting OMERO.master Windows service.
   Waiting on startup. Use CTRL-C to exit
   ...

   If you have chosen a non-default install directory (other than C:\OMERO.server), the output will look like this:

   C:\OMERO.server> bin\omero admin start
   Found default value: C:\OMERO.server\var\master
   Attempting to correct...
   Converting from C:\OMERO.server to C:\OMERO.server
   Changes made: 6
   No descriptor given. Using etc\grid\windefault.xml
   ...

   • If you would like to move the directory again, see bin\winconfig.bat --help, which gets called automatically on an initial install.

   • You can now test that you can log-in as “root”, either with the OMERO.insight client or command-line:

   C:\OMERO.server> bin\omero login
   Server: [localhost]
   Username: [root]
   Password: * root_password

JVM memory settings

The OMERO server starts a number of Java services. Memory settings for these are calculated on a system-by-system basis. An attempt has been made to have usable settings out of the box, but if you can afford to provide OMERO with more memory, it will certainly improve your overall performance. See Memory configuration on how to tune the JVM.

6.2. Server installation on Windows
OMERO.web and administration

**Note:** In order to deploy OMERO.web in a production environment such as IIS please follow the instructions under *OMERO.web deployment*.

Otherwise, the internal Django webserver can be used for evaluation and development. In this case, you need to install Django and turn debugging on, in order that static files can be served by Django:

```
C:\OMERO.server> C:\Python27\Scripts\pip.exe install -r share/web/requirements-py27-win.txt
C:\OMERO.server> bin\omero config set omero.web.application_server development
C:\OMERO.server> bin\omero config set omero.web.debug True
```

then start by

```
c:\OMERO.server> bin\omero web start
INFO:__main__:Application Starting...
INFO:root:Processing custom settings for module omeroweb.settings
...
Validating models...
0 errors found
Django version 1.8, using settings 'omeroweb.settings'
Starting development server at http://127.0.0.1:4080/
Quit the server with CTRL-BREAK.
```

**Note:** As OMERO.web 5 is based on Django 1.8, `omero.web.session_engine` and `omero.web.caches` should be unset.

```
C:\omero_dist>bin\omero config set omero.web.session_engine
C:\omero_dist>bin\omero config set omero.web.caches
```

If your deployment requires additional cache store please follow Django documentation for more details.

Once you have deployed and started the server you can use your browser to access the OMERO.web interface:

- http://localhost:4080/

![OMERO.web login](image)

Figure 6.4: OMERO.web login

---

96 [https://docs.djangoproject.com/en/1.8/releases/1.8/]
97 [https://docs.djangoproject.com/en/1.8/topics/cache/]

---

6.2. Server installation on Windows 152
**Post-installation items**

### Additional libraries

The following are optional depending on what services you require:

<table>
<thead>
<tr>
<th>Package</th>
<th>Functionality</th>
<th>Downloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Django (1.8)</td>
<td>OMERO.web</td>
<td>Django website[^98]</td>
</tr>
<tr>
<td>Pillow (2.9.0)</td>
<td>OMERO.web and Figure Export</td>
<td>Pillow[^99]</td>
</tr>
<tr>
<td>Matplotlib</td>
<td>OMERO.web</td>
<td>Matplotlib page[^100]</td>
</tr>
<tr>
<td>PyTables (2.1.0 or higher)</td>
<td>OMERO.tables</td>
<td>PyTables page[^101]</td>
</tr>
<tr>
<td>NumPy (1.2.0 or higher)</td>
<td>Scripting</td>
<td>Numpy/Scipy page[^102]</td>
</tr>
<tr>
<td>Numexpr (2.4.3 or higher)</td>
<td></td>
<td>Numexpr page[^103]</td>
</tr>
<tr>
<td>pyparsing (2.0.3 or higher)</td>
<td></td>
<td>pyparsing page[^104]</td>
</tr>
<tr>
<td>python-dateutil (2.4.2 or higher)</td>
<td></td>
<td>python-dateutil page[^105]</td>
</tr>
<tr>
<td>pytz</td>
<td></td>
<td>pytz page[^106]</td>
</tr>
<tr>
<td>six (1.9.0 or higher)</td>
<td></td>
<td>six page[^107]</td>
</tr>
<tr>
<td>virtualenv (12.1.1 or higher)</td>
<td></td>
<td>virtualenv page[^108]</td>
</tr>
</tbody>
</table>

### Backup

One of your first steps after putting your OMERO server into production should be deciding on when and how you are going to backup your database and binary data. Please do not omit this step.

### Security

You should read the [Server security and firewalls](#) page to get a good idea as to what you need to do to get OMERO clients speaking to your newly installed OMERO.server in accordance with your institution or company’s security policy.

### Advanced configuration

Once you have the base server running, you may want to try enabling some of the advanced features such as OMERO.dropbox or LDAP authentication. If you have *Flex data*, you may want to watch the [HCS configuration screencast](#). See the [Feature list](#) for more advanced features you may want to use, and [Configuration properties glossary](#) on how to get the most out of your server.

If your users are going to be importing many files in one go, for example multiple plates, you should make sure you set the maximum number of open files to a sensible level (i.e. at least 8K for production systems, 16K for bigger machines). See [Too many open files](#) for more information.

### Troubleshooting

If you encounter a problem which is not addressed by the [Troubleshooting OMERO](#) page, you can post a message to our [ome-users](#) mailing list as discussed on the [Community support](#) page. Especially the Server fails to start and Remote clients cannot connect to OMERO installation sections are a good starting point.

[^98]: https://www.djangoproject.com/
[^99]: http://pillow.readthedocs.org
[^100]: http://matplotlib.org/
[^102]: May already have been installed as a dependency of Matplot Lib.
[^103]: http://pypi.python.org/pypi/scipy
[^104]: http://pypi.python.org/pypi/numexpr
[^105]: http://pyparsing.wikispaces.com
[^106]: http://pypi.python.org/pypi/python-dateutil
[^107]: http://pytz.sourceforge.net/
[^108]: http://pypi.python.org/pypi/six
[^110]: http://cvs.openmicroscopy.org.uk/snapshots/movies/omero-4-1/mov/FlexPreview4.1-configuration.mov
[^111]: http://www.openmicroscopy.org/site/products/omero/feature-list
[^112]: http://lists.openmicroscopy.org.uk/mailman/listinfo/ome-users
OMERO diagnostics

Please include the output of the diagnostics command when asking for help with your server installation:

C:\opt\OMERO.server> bin\omero admin diagnostics

OMERO Diagnostics 5.2.6

Commands: java -version 1.8.0 (C:\Program Files\Java\jdk1.7.0_51\bin\java.EXE - 2 others)
Commands: python -V 2.7.6 (C:\Python27\python.EXE -- 2 others)
Commands: icegridnode --version 3.5.1 (c:\opt\Ice-3.5.1-win-x64-Release\bin\icegridnode.EXE)
Commands: icegridadmin --version 3.5.1 (c:\opt\Ice-3.5.1-win-x64-Release\bin\icegridadmin.EXE)
Commands: psql --version 9.3.4 (C:\Program Files\PostgreSQL\9.3\bin\psql.EXE)

Server: icegridnode running
Server: Blitz-0 active (pid = 1384, enabled)
Server: DropBox inactive (disabled)
Server: FileServer inactive (disabled)
Server: Indexer-0 active (pid = 3872, enabled)
Server: MonitorServer inactive (disabled)
Server: OMERO.Glacier2 active (pid = 5744, enabled)
Server: OMERO.IceStorm active (pid = 3520, enabled)
Server: PixelData-0 active (pid = 3096, enabled)
Server: Processor-0 inactive (disabled)
Server: Tables-0 inactive (disabled)
Server: TestDropBox inactive (enabled)
Server: OMERO.master active (running as LocalSystem)

OMERO: SSL port 4064
OMERO: TCP port 4063

Log dir: C:\opt\OMERO.server-5.0.1-ice35-b21\var\log exists

Log files: Blitz-0.log 38.0 KB
Log files: DropBox.log n/a
Log files: FileServer.log n/a
Log files: Indexer-0.log 64.0 KB errors=0 warnings=2
Log files: MonitorServer.log n/a
Log files: OMEROweb.log n/a
Log files: PixelData-0.log 3.0 KB errors=0 warnings=2
Log files: Processor-0.log n/a
Log files: Tables-0.log n/a
Log files: TestDropBox.log n/a
Log files: master.err 1.0 KB errors=2 warnings=2
Log files: master.out 0.0 KB
Log files: Total size 0.11 MB

Environment:OMERO_HOME=(unset)
Environment:OMERO_NODE=(unset)
Environment:OMERO_MASTER=(unset)
Environment:OMERO_TEMPDIR=(unset)

6.2. Server installation on Windows
Environment: PATH=C:\Program Files\PostgreSQL\9.3\bin;C:\Python27;c:\opt\Ice-3.5.1-win-x64-Release\bin;C:\Program Files\Java\jdk1.7.0_51\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\
Environment: ICE_HOME=c:\opt\Ice-3.5.1-win-x64-Release
Environment: LD_LIBRARY_PATH=(unset)
Environment: DYLD_LIBRARY_PATH=(unset)

OMERO data dir: ‘C:\opt\omerodata’ Exists? True Is writable? True
OMERO temp dir: ‘C:\Users\omero\AppData\Roaming\omero\tmp’ Exists? True Is writable? True (Size: 0)
OMERO.web status... [NOT STARTED]

Update notification

Your OMERO.server installation will check for updates each time it is started from the Open Microscopy Environment update server. If you wish to disable this functionality you should do so now as outlined on the OMERO upgrade checks page.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.2.2 OMERO.server binary repository

**Warning:** OMERO 5.2 is the last major version which will feature Windows support for OMERO.server and OMERO.web deployment. See this blog post for details. If you are installing a new server, we highly recommend you use a different OS (see Version requirements).

---

*http://blog.openmicroscopy.org/tech-issues/future-plans/deployment/2016/03/22/windows-support/

---

About

The OMERO.server binary data repository is a fundamental piece of server-side functionality. It provides optimized and indexed storage of original file, pixel and thumbnail data, attachments and full-text indexes. The repository’s directories contain various files that, together with your SQL database, constitute the information about your users and their data that OMERO.server relies upon for normal operation.

Layout

The repository is internally laid out as follows:

C:\OMERO
C:\OMERO\Pixels <<< Pixel data and pyramids
C:\OMERO\Files <<< Original file data
C:\OMERO\Thumbnails <<< Thumbnail data
C:\OMERO\FullText <<< Lucene full text search index
C:\OMERO\ManagedRepository <<< OMERO.fs filesets, with import logs
C:\OMERO\BioFormatsCache <<< Cached Bio-Formats state for rendering

Your repository is not:

- the “database”
- the directory where your OMERO.server binaries are
- the directory where your OMERO.client (OMERO.insight or OMERO.importer) binaries are
- your PostgreSQL data directory

6.2. Server installation on Windows
Locking and remote shares

The OMERO server requires proper locking semantics on all files in the binary repository. In practice, this means that remotely mounted shares such as AFS, CIFS, and NFS can cause issues. If you have experience and/or the time to manage and monitor the locking implementations of your remote filesystem, then using them as for your binary repository should be fine.

If, however, you are seeing errors such as NullPointerExceptions, “Bad file descriptors” and similar in your server log, then you will need to use directly connected disks.

**Warning:** If your binary repository is a remote share and mounting the share fails or is dismounted, OMERO will continue operating using the mount point instead! To prevent this, make the mount point read-only for the OMERO user so that no data can be written to the mount point.

Repository location

**Note:** It is strongly recommended that you make all changes to your OMERO binary repository with the server shut down. Changing the `omero.data.dir` configuration does not move the repository for you, you must do this yourself. Remember that `C:\` style paths must have backslashes escaped. We strongly discourage the use of network mapped drives as locations for either the binary repository or the OMERO server installation.

Your repository location can be changed from its `C:\OMERO` default by modifying your OMERO server configuration as follows:

```
C:\> cd C:\OMERO.server
C:\OMERO.server\> bin\omero config set omero.data.dir D:\OMERO
```

The suggested procedure is to shut down your OMERO server instance, move your repository, change your `omero.data.dir` and then start the instance back up. For example:

```
C:\> cd C:\OMERO.server
C:\OMERO.server\> bin\omero admin stop
C:\OMERO.server\> move C:\OMERO D:\
C:\OMERO.server\> bin\omero config set omero.data.dir D:\OMERO
C:\OMERO.server\> bin\omero admin start
```

The `omero.managed.dir` property for the OMERO.fs managed repository may be adjusted similarly, even to a directory outside `omero.data.dir`.

**Note:** The managed repository should be located and configured to allow the OMERO server processes fast access to the uploaded files sets that it contains.

Access permissions

Your repository should be owned or accessible by the same user that is starting your OMERO server instance which may be different from the user you use to start OMERO. See `OMERO.server Windows Service` for more information.

To modify the access permissions to the binary repository, the OMERO folder properties can be accessed and the permissions settings changed (see `Repository Folder Permissions`). Another option (useful for batch permission changes) is the `icacls` (Windows 7)/`cacls` (Windows XP) command line utility. Please note that the new permissions will appear as Special Permissions in the Security tab when viewing folder properties. An example invocation allowing the user `omeservice` to read (`R`) and write (`W`) from and to the OMERO directory:

```
C:\>icacls OMERO /grant omeservice:RW
processed file: OMERO
Successfully processed 1 files; Failed processing 0 files
```
Repository size

At minimum, the binary repository should be comfortably larger than the images and other files that users may be uploading to it. It is fine to set `omero.data.dir` or `omero.managed.dir` to very large volumes, or to use logical volume management to conveniently increase space as necessary.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.2.3 OMERO.server and PostgreSQL

**Warning:** OMERO 5.2 is the last major version which will feature Windows support for OMERO.server and OMERO.web deployment. See this blog post[^1] for details. If you are installing a new server, we highly recommend you use a different OS (see Version requirements).

[^1]: http://blog.openmicroscopy.org/tech-issues/future-plans/deployment/2016/03/22/windows-support/

In order to be installed, OMERO.server requires a running PostgreSQL instance that is configured to accept connections over TCP. This section explains how to ensure that you have the correct PostgreSQL version and that it is installed and configured correctly. For Windows-specific installation instructions, first see OMERO.server installation.

Ensuring you have a valid PostgreSQL version

For OMERO 5.2, PostgreSQL version 9.4 or later is required; version 9.4 is recommended. Make sure you are using a supported version[^2].

[^2]: http://www.postgresql.org/support/versioning/
If your existing PostgreSQL installation is version 9.3 or earlier, it is recommended that you upgrade to a more up-to-date version. Before upgrading, stop the OMERO server and then perform a full dump of the database using `pg_dump`. See the OMERO.server backup and restore section for further details.

If a PostgreSQL server was not provided by your system, EnterpriseDB\(^\text{114}\) provide an installer.

**Checking PostgreSQL port listening status**

You can check if PostgreSQL is listening on the default port (TCP/5432) by running the following command:

```
C:\> netstat -an | find "5432"
```

**Note:** The exact output of this command will vary. The important thing to recognize is whether or not a process is listening on TCP/5432.

If you cannot find a process listening on TCP/5432 you will need to find your `postgresql.conf` file and enable PostgreSQL’s TCP listening mode. The exact location of the `postgresql.conf` file varies between installations.

Once you have found the location of the `postgresql.conf` file on your particular installation, you will need to enable TCP listening. The area of the configuration file you are concerned about should look similar to this:

```
#listen_addresses = 'localhost'  # what IP address(es) to listen on;
# comma-separated list of addresses;
# defaults to 'localhost', '*' = all
#port = 5432
max_connections = 100
# note: increasing max_connections costs ~400 bytes of shared memory per
# connection slot, plus lock space (see max_locks_per_transaction). You
# might also need to raise shared_buffers to support more connections.
#superuser_reserved_connections = 2
#unix_socket_directory = *
#unix_socket_group = *
#unix_socket_permissions = 0777  # octal
#bonjour_name = *  # defaults to the computer name
```

**PostgreSQL HBA (host based authentication)**

OMERO.server must have permission to connect to the database that has been created in your PostgreSQL instance. This is configured in the host based authentication file, `pg_hba.conf`. Check the configuration by examining the contents of `pg_hba.conf`. It’s important that at least one line allows connections from the loopback address (127.0.0.1) as follows:

```
# TYPE   DATABASE   USER    CIDR-ADDRESS    METHOD
# IPv4   local connections:
host   all   all   127.0.0.1/32   md5
# IPv6   local connections:
host   all   all   ::1/128   md5
```

**Note:** The other lines that are in your `pg_hba.conf` are important either for PostgreSQL internal commands to work or for existing applications you may have. Do not delete them.

See also:

- PostgreSQL\(^\text{115}\) Interactive documentation for the current release of PostgreSQL.
- Connections and Authentication\(^\text{116}\) Section of the PostgreSQL documentation about configuring the server using `postgresql.conf`.

\(^\text{114}\)http://www.enterprisedb.com/
OMERO Documentation, Release 5.2.6

Client Authentication

Chapter of the PostgreSQL documentation about configuring client authentication with pg_hba.conf.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.2.4 OMERO.web deployment

Warning: OMERO 5.2 is the last major version which will feature Windows support for OMERO.web deployment. See this blog post for details. If you are deploying a new installation of OMERO.web, we highly recommend you use a different OS (see Version requirements). The guidance below applies to OMERO 5.2. Code fixes for OMERO.web deployment on Windows have not been back-ported to the 5.1 line and this guide cannot be used to deploy OMERO.web 5.1.x on Windows.

OMERO.web is the web application component of the OMERO platform which allows for the management, visualization (in a fully multi-dimensional image viewer) and annotation of images from a web browser. It also includes the ability to manage users and groups.

OMERO.web is an integral part of the OMERO platform and can be deployed with:

- IIS 6.0 or 7.0 on Microsoft Windows
- WSGI using a WSGI capable web server such as Apache 2.2+ (with mod wsgi enabled).
- The built-in Django lightweight development server (for testing only; see the OMERO.web deployment for developers page)

If you need help configuring your firewall rules, see the Server security and firewalls page.

Prerequisites

- OMERO and its prerequisites (see OMERO.server installation).
- Python 2.7
  - Pillow should be available for your distribution. We currently do not support version 3.0+.
  - Matplotlib should be available for your distribution
- IIS or WSGI capable web server

Deployment (Windows)

See Customizing your OMERO.web installation for additional customization options.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

OMERO.web IIS deployment (Windows)

Warning: OMERO 5.2 is the last major version which will feature Windows support for OMERO.server and OMERO.web deployment. See this blog post for details. If you are installing a new server, we highly recommend you use a different OS (see Version requirements).

http://blog.openmicroscopy.org/tech-issues/future-plans/deployment/2016/03/22/windows-support/

---

117 http://wsgi.readthedocs.org/en/latest/
119 http://httpd.apache.org
120 http://modwsgi.readthedocs.org/en/develop/
121 http://matplotlib.org/
122 http://blog.openmicroscopy.org/tech-issues/future-plans/deployment/2016/03/22/windows-support/
IIS configuration (Windows)

**Note:** Since OMERO 5.2, the OMERO.web framework no longer bundles a copy of the Django package, instead manual installation of the Django dependency is required. It is highly recommended to use Django 1.8\(^{123}\) on Windows. For more information see Python on the Version requirements page.

Install Django 1.8\(^{124}\) using package requirements file:

C:\> pip install -r C:\OMERO.server\share\web\requirements-py27-win.txt

**Note:** For more details refer to Django 1.8 installation on Windows\(^{125}\).

Install additional Python libraries:

Download the following whl files then run:

```
set PYTHONPIP=C:\Python27\Scripts\pip.exe

%PYTHONPIP% install matplotlib-1.4.3-cp27-none-win_amd64.whl
%PYTHONPIP% install numpy-1.9.2+mkl-cp27-none-win_amd64.whl
%PYTHONPIP% install numexpr-2.4.3-cp27-none-win_amd64.whl
%PYTHONPIP% install Pillow-2.8.1-cp27-none-win_amd64.whl
%PYTHONPIP% install pyparsing-2.0.3-py2-none-any.whl
%PYTHONPIP% install python_dateutil-2.4.2-py2.py3-none-any.whl
%PYTHONPIP% install pytz-2015.2-py2.py3-none-any.whl
%PYTHONPIP% install six-1.9.0-py2.py3-none-any.whl
%PYTHONPIP% install tables-3.1.1-cp27-none-win_amd64.whl
%PYTHONPIP% install virtualenv-12.1.1-py2.py3-none-any.whl
%PYTHONPIP% install pywin32-219-cp27-none-win_amd64.whl
C:\Python27\Scripts\pywin32_postinstall.py -install
```

A straightforward set of steps is required to get the ISAPI WSGI\(^{126}\) handler for OMERO.web working with your IIS deployment.

- Ensure that the ISAPI for IIS options are installed.
- Download and install an ISAPI WSGI\(^{127}\). If you download the Source Distribution (should be compatible with all versions of Windows/IIS) unzip the archive and run:

  ```
  python setup.py install
  ```

  Alternatively if you are using a 32-bit system you can use the Windows Installer.
  
  - For extended compatibility with multiple IIS versions ISAPI WSGI uses the IIS 6 WMI interface to interact with your IIS deployment. If you are using IIS 7 you must enable the IIS 6 WMI backwards compatibility options. Also make sure to install the ISAPI extensions and filters, as shown in the figure below. For further information see IIS Documentation - ISAPI Filters\(^{128}\) respectively IIS Documentation - ISAPI CGI Restrictions\(^{129}\).

- The Advanced Settings... for the application pool assigned to your default site require modification using the IIS Manager as follows:
  
  - **Idle Time-out (minutes)** - to stop the worker process from suspending during inactivity periods, this setting needs to be changed to 0.

- To avoid rendering errors on OMERO.web it is recommended to add the custom HTTP response header in the Web site pane.

\(^{123}\)https://docs.djangoproject.com/en/1.8/releases/1.8/

\(^{124}\)https://docs.djangoproject.com/en/1.8/releases/1.8/

\(^{125}\)https://docs.djangoproject.com/en/1.8/releases/1.8/

\(^{126}\)http://code.google.com/p/isapi-wsgi/

\(^{127}\)http://code.google.com/p/isapi-wsgi/downloads/list

\(^{128}\)https://www.iis.net/configreference/system.webserver/isapifilters

\(^{129}\)https://www.iis.net/configreference/system.webserver/security/isapicgirestriction

6.2. Server installation on Windows
- Make sure that OMERO’s log directory has full read/write permissions for all users
- Configure OMERO.web bindings for IIS

C:\OMERO.server> bin\omero web iis

This will automatically add two applications, /omero and /static, to the default web site and application pool. You can customize these locations by setting `omero.web.prefix` and `omero.web.static_url`.

**Note:** You can make manual changes to the IIS OMERO configuration but they may be lost when OMERO.server is upgraded.

**Note:** As OMERO.web 5 is based on Django 1.8\(^{130}\), `omero.web.session_engine` and `omero.web.caches` should be unset.

\(^{130}\)https://docs.djangoproject.com/en/1.8/releases/1.8/
If your deployment requires additional cache store please follow the Django documentation\footnote{https://docs.djangoproject.com/en/1.8/topics/cache/} for more details.

If you wish to use Apache or Nginx please follow the Unix instructions

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

**DEPRECATED: OMERO.web Apache and mod_wsgi deployment (Unix/Linux)**

Apache 2.2+ with mod_wsgi configuration (Unix/Linux)

**Note:** Since OMERO 5.2, the OMERO.web framework no longer bundles a copy of the Django package, instead manual installation of the Django dependency is required. It is highly recommended to use Django 1.8\footnote{https://docs.djangoproject.com/en/1.8/releases/1.8/} (LTS) which requires Python 2.7. For more information see Python on the Version requirements page.

Also note that support for Apache deployment is deprecated as of OMERO 5.2.6 and is likely be dropped during the 5.3.x line.

OMERO.web uses the Web Server Gateway Interface (WSGI)\footnote{http://wsgi.readthedocs.org/en/latest/learn.html} as a primary deployment platform. The Python standard PEP 3333\footnote{https://www.python.org/dev/peps/pep-3333/} that describes how a web server communicates with web applications.

Apache default configuration (Unix/Linux)  
Install Django 1.8\footnote{https://docs.djangoproject.com/en/1.8/releases/1.8/} using package requirements file:

```bash
$ pip install -r share/web/requirements-py27-apache.txt
```

**Note:** For more details refer to how to install Django 1.8\footnote{https://docs.djangoproject.com/en/1.8/topics/install/#install-the-django-code} or how to upgrade Django to 1.8\footnote{https://docs.djangoproject.com/en/1.8/topics/install/#remove-any-old-versions-of-django}.

Additional settings can be configured by changing the following properties:

- `omero.web.application_server.max_requests` to 500
- `omero.web.wsgi_workers` to `(2 \times \text{NUM\_CORES}) + 1`

6.2. Server installation on Windows
Note: Do not scale the number of workers to the number of clients you expect to have. OMERO.web should only need 4-12 worker processes to handle many requests per second.

Apache configuration (Unix/Linux) If you have installed Apache, install mod_wsgi\(^\text{138}\).

OMERO can automatically generate a configuration file for your web server. The location of the file will depend on your system, please refer to your web server’s manual. See Customizing your OMERO.web installation for additional customization options.

Set the following:

```
$bin/omero config set omero.web.application_server "wsgi"
```

Creates symlinks for static media files

\(^{138}\text{http://www.modwsgi.org/}\)
$bin/omero web syncmedia

To create a site configuration file for inclusion in the main Apache configuration redirect the output of the following command into a file:

For Apache 2.4, run the following command:

$ bin/omero web config apache24 > httpd-omero.conf

For Apache 2.2, run the following command:

$ bin/omero web config apache > httpd-omero.conf

Configuration file:

<VirtualHost _default_:80>
  WSGIDaemonProcess omeroweb processes=5 threads=1 maximum-requests=0 display-name=%{GROUP} user=omero
  WSGIScriptAlias / /home/omero/OMERO.server/lib/python/omeroweb/wsgi.py process-group=omeroweb
  <Directory "/home/omero/OMERO.server/lib/python/omeroweb">
    WSGIProcessGroup omeroweb
    WSGIApplicationGroup %{GLOBAL}
    Order allow,deny
    Allow from all
  </Directory>
  Alias /static /home/omero/OMERO.server/lib/python/omeroweb/static
  <Directory "/home/omero/OMERO.server/lib/python/omeroweb/static">
    Options -Indexes FollowSymLinks
    Order allow,deny
    Allow from all
  </Directory>
</VirtualHost>

# see https://code.google.com/p/modwsgi/wiki/ConfigurationIssues
WSGISocketPrefix run/wsgi
# Use this on Ubuntu/Debian/MacOSX systems:
# WSGISocketPrefix /var/run/wsgi

To configure an HTTPS server follow the Apache documentation\(^{139}\).

Then reload Apache.

Running OMERO.web (Unix/Linux)  OMERO.web is used in ’daemon’ mode where UNIX sockets are used to communicate between the Apache child processes and the daemon processes which are to handle a request. OMERO.web does not provide any management for these ‘daemon’ processes; bin/omero web start/status/stop will not work as they are for managing gunicorn processes only.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

\(^{139}\)http://httpd.apache.org/docs/trunk/mod/mod_ssl.html

6.2. Server installation on Windows
OMERO.web Nginx and Gunicorn deployment (Unix/Linux)

Note: Since OMERO 5.2, the OMERO.web framework no longer bundles a copy of the Django package, instead manual installation of the Django dependency is required. It is highly recommended to use Django 1.8 (LTS) which requires Python 2.7. For more information see Python on the Version requirements page.

OMERO.web uses the Web Server Gateway Interface (WSGI) as a primary deployment platform. The Python standard PEP that describes how a web server communicates with web applications.

Gunicorn default configuration (Unix/Linux)  Install Django 1.8 and Gunicorn >= 19.3 using the package requirements file:

$ pip install -r share/web/requirements-py27-nginx.txt

Note: For more details refer to how to install Django 1.8 or upgrade Django to 1.8.

Additional settings can be configured by changing the following properties:

- omero.web.application_server.max_requests to 500
- omero.web.wsgi_workers to (2 x NUM_CORES) + 1

Note: Do not scale the number of workers to the number of clients you expect to have. OMERO.web should only need 4-12 worker processes to handle many requests per second.

- omero.web.wsgi_args Additional arguments. For more details check Gunicorn Documentation.

Nginx configuration (Unix/Linux)  If you have installed Nginx, OMERO can automatically generate a configuration file for your web server. The location of the file will depend on your system, please refer to your web server’s manual. See Customizing your OMERO.web installation for additional customization options.

To create a site configuration file for inclusion in a system-wide nginx configuration redirect the output of the following command into a file:

$ bin/omero web config nginx

upstream omeroweb {
    server 127.0.0.1:4080 fail_timeout=0;
}

server {
    listen 80;
    server_name $hostname;
    sendfile on;
    client_max_body_size 0;
    # maintenance page serve from here
    location @maintenance {

140https://docs.djangoproject.com/en/1.8/releases/1.8/
142https://www.python.org/dev/peps/pep-3333/
143https://docs.djangoproject.com/en/1.8/releases/1.8/
144http://gunicorn.org
145https://docs.djangoproject.com/en/1.8/topics/install/#install-the-django-code
146https://docs.djangoproject.com/en/1.8/topics/install/#remove-any-old-versions-of-django

6.2. Server installation on Windows
root /home/omero/OMERO.server/etc/templates/error;
try_files $uri /maintainance.html =502;
}

# weblitz django apps serve media from here
location /static {
  alias /home/omero/OMERO.server/lib/python/omeroweb/static;
}

location @proxy_to_app {
  proxy_set_header X-Forwarded-Proto $scheme;
  proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
  proxy_set_header Host $http_host;
  proxy_redirect off;
  proxy_buffering off;

  proxy_pass http://omeroweb;
}

location / {
  error_page 502 @maintenance;
  # checks for static file, if not found proxy to app
  try_files $uri @proxy_to_app;
}

Note: OMERO.web requires body_in_file_only adjusted in your default nginx config because nginx must buffer incoming data. Make sure you have that set to the following config:

http {
...
  sendfile on;
  send_timeout 60s;
  client_max_body_size 0;
  ...
}

To configure an HTTPS server follow the nginx documentation\(^{148}\).

Running OMERO.web (Unix/Linux)  Start the Gunicorn worker processes listening by default on 127.0.0.1:4080:

$ bin/omero web start
... static files copied to '/home/omero/OMERO.server/lib/python/omeroweb/static'.
Starting OMERO.web... [OK]

The Gunicorn workers are managed separately from other OMERO.server processes. You can check their status or stop them using the following commands:

$ bin/omero web status
OMERO.web status... [RUNNING] (PID 59217)
$ bin/omero web stop
Stopping OMERO.web... [OK]
Django WSGI workers (PID 59217) killed.


6.2. Server installation on Windows 166
Download limitations  In order to offer users the ability to download data from OMERO.web you have to deploy using EXPERIMENTAL: Async workers. OMERO.web is able to handle multiple clients on a single worker thread switching context as necessary while streaming binary data from OMERO.server. Depending on the traffic and scale of the repository you should configure connections and speed limits on your server to avoid blocking resources. We recommend you run benchmark and performance tests. It is also possible to apply Download restrictions and offer alternative access to binary data.

Note: Handling streaming request/responses requires proxy buffering to be turned off. For more details refer to Gunicorn deployment\(^\text{149}\) and Nginx configuration\(^\text{150}\).

Note: omero.web.application_server.max_requests should be set to 0

Benchmark  We run example benchmarks on rendering thumbnails and 512x512 pixels planes for 100 concurrent users making 5000 requests in total:

$ ab -n 5000 -c 100 https://server.openmicroscopy.org/omero/webclient/render_thumbnail/size/96/1234/
This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Server Software: nginx/1.9.9
Server Hostname: server.openmicroscopy.org
Server Port: 80

Document Path: /omero/webclient/render_thumbnail/size/96/31851/
Document Length: 1880 bytes

Concurrency Level: 100
Time taken for tests: 224.488 seconds
Complete requests: 5000
Failed requests: 0
Write errors: 0
Total transferred: 10450000 bytes
HTML transferred: 9400000 bytes
Requests per second: 22.27 [#/sec] (mean)
Time per request: 4489.763 [ms] (mean)
Time per request: 44.898 [ms] (mean, across all concurrent requests)
Transfer rate: 45.46 [Kbytes/sec] received

Connection Times (ms)

<table>
<thead>
<tr>
<th></th>
<th>min</th>
<th>mean[+/-sd]</th>
<th>median</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td>0</td>
<td>0 [0.3]</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Processing</td>
<td>435</td>
<td>4446 685.2</td>
<td>4363</td>
<td>7644</td>
</tr>
<tr>
<td>Waiting</td>
<td>432</td>
<td>4446 685.3</td>
<td>4362</td>
<td>7644</td>
</tr>
<tr>
<td>Total</td>
<td>435</td>
<td>4446 685.1</td>
<td>4363</td>
<td>7644</td>
</tr>
</tbody>
</table>

Percentage of the requests served within a certain time (ms)

<table>
<thead>
<tr>
<th>Level</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50%</td>
<td>4363</td>
</tr>
<tr>
<td>66%</td>
<td>4553</td>
</tr>
<tr>
<td>75%</td>
<td>4670</td>
</tr>
<tr>
<td>80%</td>
<td>4750</td>
</tr>
<tr>
<td>90%</td>
<td>5072</td>
</tr>
<tr>
<td>95%</td>
<td>5398</td>
</tr>
<tr>
<td>98%</td>
<td>6795</td>
</tr>
<tr>
<td>99%</td>
<td>6955</td>
</tr>
<tr>
<td>100%</td>
<td>7644 (longest request)</td>
</tr>
</tbody>
</table>

\(^{150}\)http://nginx.org/en/docs/nginx_http_proxy_module.html#proxy_buffering
$ ab -n 5000 -c 100 http://server.openmicroscopy.org/omero/webclient/render_image/1234/20/0/
This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/

Server Software: nginx/1.9.9
Server Hostname: server.openmicroscopy.org
Server Port: 80

Document Path: /omero/webclient/render_image/1234/20/0/
Document Length: 24293 bytes

Concurrency Level: 100
Time taken for tests: 247.154 seconds
Complete requests: 5000
Failed requests: 0
Write errors: 0
Total transferred: 122515000 bytes
HTML transferred: 121465000 bytes
Requests per second: 20.23 [#/sec] (mean)
Time per request: 4943.080 [ms] (mean)
Time per request: 49.431 [ms] (mean, across all concurrent requests)
Transfer rate: 484.09 [Kbytes/sec] received

Connection Times (ms)

<table>
<thead>
<tr>
<th></th>
<th>min</th>
<th>mean[+-sd]</th>
<th>median</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
<td>0</td>
</tr>
<tr>
<td>Processing</td>
<td>482</td>
<td>4898 855.3</td>
<td>4737</td>
<td>8303</td>
</tr>
<tr>
<td>Waiting</td>
<td>476</td>
<td>4898 855.3</td>
<td>4737</td>
<td>8303</td>
</tr>
<tr>
<td>Total</td>
<td>482</td>
<td>4898 855.2</td>
<td>4737</td>
<td>8303</td>
</tr>
</tbody>
</table>

Percentage of the requests served within a certain time (ms)

<table>
<thead>
<tr>
<th></th>
<th>50%</th>
<th>66%</th>
<th>75%</th>
<th>80%</th>
<th>90%</th>
<th>95%</th>
<th>98%</th>
<th>99%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4737</td>
<td>5041</td>
<td>5250</td>
<td>5397</td>
<td>5862</td>
<td>6621</td>
<td>7301</td>
<td>8062</td>
<td>8303</td>
</tr>
</tbody>
</table>

Troubleshooting In order to identify why OMERO.web is not available run:

$ bin/omero web status

Then consult nginx error.log and OMERO.server/var/log/OMEROweb.log

For more details check OMERO.web issues.

Debugging To run the WSGI server in debug mode, enable Gunicorn logging using omero.web.wsgi_args:

$ bin/omero config set omero.web.wsgi_args --log-level=DEBUG --error-logfile=/home/omero/OMERO.server/var/log/error.log

EXPERIMENTAL: Gunicorn advance configuration (Unix/Linux)

Note: Experimental configurations are not ready for production use. Configuration may change. Features may not work properly.


6.2. Server installation on Windows
OMERO.web deployment can be configured with sync and async workers. Sync workers are faster and recommended for a data repository with Download restrictions. If you wish to offer users the ability to download data then you have to use async workers; read more about Download limitations.

For more details refer to Gunicorn design\(^\text{152}\).

**EXPERIMENTAL: Sync workers**

**Note:** Experimental configurations are not ready for production use. Configuration may change. Features may not work properly.

- Install futures\(^\text{153}\)

  ```
  $ pip install futures
  ```

  Additional settings can be configured by changing the following properties:

  - The number of worker threads for handling requests, see Gunicorn threads\(^\text{154}\)

    ```
    $ bin/omero config set omero.web.wsgi_worker_class
    $ bin/omero config set omero.web.wsgi_threads $(2-4 x NUM_CORES)
    ```

**EXPERIMENTAL: Async workers**

**Note:** Experimental configurations are not ready for production use. Configuration may change. Features may not work properly.

- Install Gevent > = 0.13\(^\text{155}\)

  ```
  $ pip install "gevent>=0.13"
  ```

  Additional settings can be configured by changing the following properties:

  - The maximum number of simultaneous clients, see Gunicorn worker-connections\(^\text{156}\)

    ```
    $ bin/omero config set omero.web.wsgi_worker_class gevent
    $ bin/omero config set omero.web.wsgi_worker_connections 1000
    $ bin/omero config set omero.web.application_server.max_requests 0
    ```

**Logging in to OMERO.web**

Once you have deployed and started the server, you can use your browser to access the OMERO.web client:

- [http://your_host/omero](http://your_host/omero) OR, for development server: [http://localhost:4080](http://localhost:4080)

**OMERO.web Maintenance (Windows)**

- Session cookies omero.web.session_expire_at_browser_close:

  - A boolean that determines whether to expire the session when the user closes their browser. See Django Browser-length sessions vs. persistent sessions documentation\(^\text{157}\) for more details. Default: True.

    ```
    C:\omero_dist>bin\omero config set omero.web.session_expire_at_browser_close "True"
    ```

---

\(^{152}\)[http://docs.gunicorn.org/en/stable/design.html]

\(^{153}\)[https://pypi.python.org/pypi/futures]


\(^{155}\)[http://www.gevent.org/]


\(^{157}\)[https://docs.djangoproject.com/en/1.8/topics/http/sessions/#browser-length-vs-persistent-sessions]
The age of session cookies, in seconds. Default: 86400.

```bash
C:\omero_dist>bin\omero config set omero.web.session_cookie_age 86400
```

- Session engine:
  - Each session for a logged-in user in OMERO.web is kept in the session store. Stale sessions can cause the store to grow with time. OMERO.web uses by default the OS file system as the session store backend and does not automatically purge stale sessions, see Django file-based session documentation\(^{158}\) for more details. It is therefore the responsibility of the OMERO administrator to purge the session cache using the provided management command:

```bash
C:\omero_dist>bin\omero web clearsessions
```

It is recommended to call this command on a regular basis, for example as a daily cron job, see Django clearing the session store documentation\(^{159}\) for more information.

**Note:** OMERO.web offers alternative session backends to automatically delete stale data using the cache session store backend, see Django cached session documentation\(^{160}\) for more details. After installing all the cache prerequisites set the following:

```bash
C:\omero_dist>bin\omero config set omero.web.caches '{"default": { "BACKEND": "django.core.cache.backends.memcached.MemcachedCache", "LOCATION": "127.0.0.1:11211", "TIMEOUT": "86400" } }
C:\omero_dist>bin\omero config set omero.web.session_engine django.contrib.sessions.backends.cache
```

---

**Customizing your OMERO.web installation**

All configuration options can be found on various sections of Web developers documentation. For the full list, refer to Web properties or:

```bash
C:\omero_dist>bin\omero web -h
```

The most popular configuration options include:

- Debug mode, see omero.web.debug.
- Customize login page with your own logo, see omero.web.login_logo.
- Customizing index page, see omero.web.index_template.

\(^{158}\)https://docs.djangoproject.com/en/1.8/topics/http/sessions/#using-file-based-sessions

\(^{159}\)https://docs.djangoproject.com/en/1.8/topics/http/sessions/#clearing-the-session-store

\(^{160}\)https://docs.djangoproject.com/en/1.8/topics/http/sessions/#using-cached-sessions
Enabling a public user see Public data in the repository.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.2.5 OMERO.server Windows Service

**Warning:** OMERO 5.2 is the last major version which will feature Windows support for OMERO.server and OMERO.web deployment. See this blog post for details. If you are installing a new server, we highly recommend you use a different OS (see Version requirements).

OMERO.server installs a Windows Service to make the startup of the software automatic at Windows boot time.

The `omero admin start` command creates and starts a Windows service. In turn, `omero admin stop` stops and deletes the OMERO Windows service. Therefore, once `omero admin start` succeeds, it is possible to use all the regular Windows utilities, like `sc.exe` or the Services Manager, to stop OMERO.server without having the service removed completely.

If required, the OMERO.server service can be run as a different user (by modifying the Log On settings of the Windows service).

Service Log On user

Default Windows Local System user

When no specific Windows user has been defined using `omero config set`, the OMERO.server starts as the Local System user. This user has enough permissions to access data on the local file system. In most circumstances that should allow the OMERO.server service to access data inside the binary repository.

Custom user

A custom user can be configured to run the OMERO.server service. You can configure the OMERO Windows user by setting `omero.windows.user` and `omero.windows.pass`:

```
C:\OMERO.server\> bin\omero config set omero.windows.user USERNAME
C:\OMERO.server\> bin\omero config set omero.windows.pass PASSWORD
```

**Warning:** Setting `omero.windows.pass` exposes your user password in the OMERO configuration.

The user credentials can also be specified on the command line when running `omero admin start`. The `-u` parameter value is the user name, while the value of `-w` corresponds to the user’s password:

```
C:\OMERO.server\> bin\omero admin start -u omeservice -w password
```

You can verify that a different user has been set as the Log On user for the OMERO.server service by accessing the Windows Services Manager (see Windows Service Log On User Settings).

Service startup mode

To start the Services Manager, simply navigate to `Start → All Programs → Accessories → Run` (Windows 7). In the dialog, type in `services.msc` and select `OK` (see Run Windows Services Manager).

This will bring up the Windows Services Manager. Here you can see the OMERO.master service running and also stop it. Additionally the Log On tab can be accessed here to configure under which user name the service is started (see OMERO.server binary repository).
It is also possible to change the service start-up type from Automatic to Manual. The automatic mode guarantees that OMERO.server will start during the Windows boot phase. Manual mode allows the logged in user or administrator to start the service after Windows has finished booting.

**Service events**

Windows Event Viewer allows for watching events occurring in the OMERO.server service. To start the viewer, follow the same path as for Windows Services Manager, but this time type in `eventvwr.msc` (see *Starting Event viewer*).

The status events from OMERO.master will be registered in the Application view (though the log output from the server is in the configured directory).

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.
6.3 Command Line Interface as an OMERO admin tool

When first beginning to work with the OMERO server, the `omero db`, `omero config`, and `omero admin` commands will be the first you will need.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.3.1 Database tools

Rather than try to provide the functionality of a RDBM tool like `psql`, the `omero db script` command helps to generate SQL scripts for building your database. You can then use those scripts from whatever tool is most comfortable for you:

```bash
$ bin/omero db script --password secretpassword
```
Using OMERO5.2 for version
Using 0 for patch
Using password from commandline
Saving to /home/omero/OMERO5.2__0.sql

If you do not specify the OMERO root password on the command line you will be prompted to enter it.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 6.3.2 Server configuration

The `omero config` command is responsible for reading / writing user-specific profiles stored under `etc/grid/config.xml`. To get the current profile, use the `omero config def` command:

```bash
$ bin/omero config def default
```

You can then examine the current profile keys using `omero config get` and set key-value pairs using `omero config set`:

```bash
$ bin/omero config get
$ bin/omero config set example "my first value"

You can use the `OMERO_CONFIG` environment variable to point at a different profile, e.g.:

```bash
$ OMERO_CONFIG=another bin/omero config def another
$ OMERO_CONFIG=another bin/omero config get
$ OMERO_CONFIG=another bin/omero config set example "my second value"
```
The values set via `omero config set` override those compiled into the server jars. The default values which are set can be seen in the Configuration properties glossary. To add several values to a configuration, you can pipe them via standard in using `omero config load`:

```
$ grep omero.ldap etc/omero.properties | OMERO_CONFIG=ldap bin/omero config load
```

Each of these values can then be modified to suit your local setup. To remove one of the key-value pairs, pass no second argument:

```
$ OMERO_CONFIG=ldap bin/omero config set omero.ldap.trustStore
$ OMERO_CONFIG=ldap bin/omero config set omero.ldap.trustStorePassword
$ OMERO_CONFIG=ldap bin/omero config set omero.ldap.keyStore
$ OMERO_CONFIG=ldap bin/omero config set omero.ldap.keyStorePassword
```

If you will be using a particular profile more frequently you can set it as your default using the `omero config def` command:

```
$ bin/omero config def ldap
```

And finally, if you would like to remove a profile, for example to wipe a given password off of a system, use `omero config drop`:

```
$ bin/omero config drop
```

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.
6.3.3 Server administration

Server start

Once your database has been properly configured and your config profile is set to use that database, you are ready to start your server using the `omero admin start` command:

```
$ bin/omero admin start
```

This command performs the following operations in order:

1. rewrites the configuration files if `--force-rewrite` is passed or the server has never been started
2. checks the server status, i.e. pings the master node using the IceGrid administration tool
3. aborts the command if a server is running
4. rewrites the configuration files if it has not been done at step 1
5. starts the server
6. waits until the server is up

Most configuration files under `etc/grid` are generated using the templates under `etc/grid/templates` and the server configuration stored in `etc/grid/config.xml`. The rewriting step updates the JVM memory settings (see Memory configuration) and the server ports (see Ports) based on the server configuration.

`-h, --help`
Display the help for this subcommand.

`--foreground`
This option is particularly useful for debugging and maintenance and allows for starting the server up in the foreground, that is without creating a daemon on UNIX-like systems or service on Windows. During the lifetime of the server, the prompt from which it was launched will be blocked.

`--force-rewrite`
This option forces the server configuration files under `etc/grid` to be rewritten before the status of the server is checked.

Server stop

To stop a running server, you can invoke the `omero admin stop` subcommand:

```
$ bin/omero admin stop
```

This command does the following operations in order:

1. rewrites the server configuration files if `--force-rewrite` is passed
2. checks the server status, i.e. pings the master node using the IceGrid administration tool
3. aborts the command if no server is running
4. stops the server
5. waits until the server is down

`-h, --help`
Display the help for this subcommand.

`--force-rewrite`
This option forces the configuration files to be rewritten before the server status is checked.
Server restart

To stop and start the server in a single command, you can use the `omero admin restart` command:

```bash
$ bin/omero admin restart
```

The `restart` subcommand supports the same options as `omero admin start`.

Server diagnostics

To debug a server or inspect the configuration, you can use the `omero admin diagnostics` command:

```bash
$ bin/omero admin diagnostics
```

The output of this command will report information about:

- the server prerequisites (`psql`, `java`)
- the server environment variables
- the server memory settings and ports
- the status of the binary repository

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.3.4 User/group management

The `omero user` and `omero group` commands provide functionalities to add and manage users and groups on your database.

User creation

New users can be added to the database using the `omero user add` command:

```bash
$ bin/omero user add -h
```

During the addition of the new user, you will need to specify the first and last name of the new user and their username as well as the groups the user belongs to. To add John Smith as a member of group 2 identified as jsmith, enter:

```bash
$ bin/omero user add jsmith John Smith 2
```

Additional parameters such as the email address, institution, middle name etc can be passed as optional arguments to the `omero user add` command.

If you are using ldap as an authentication backend, you can create an OMERO user account for jsmith using the `omero ldap create` command, which allows the administrator to add jsmith to an OMERO group, before they have ever logged in to OMERO:

```bash
$ bin/omero ldap create jsmith
```

Converting non-LDAP users to LDAP authentication

If you want to take an existing (non-LDAP) user and ‘upgrade’ them to using LDAP you can do so using the `omero ldap setdn` command:
$ bin/omero ldap setdn -h

The process is also reversible so that the OMERO password for a user rather than the LDAP password will be used. See the caveat in the setdn help output below:

choice

Enable LDAP login for user (admins only)

Once LDAP login is enabled for a user, the password set via OMERO is ignored, and any attempt to change it will result in an error. When you disable LDAP login, the previous password will be in effect, but if the user never had a password, one will need to be set!

Positional Arguments:
choice Enable/disable LDAP login (true/false)

Optional Arguments:

In addition to any higher level options

-h, --help show this help message and exit
--user-id USER_ID ID of the user.
--user-name USER_NAME Name of the user.
--group-id GROUP_ID ID of the group.
--group-name GROUP_NAME Name of the group.

Login arguments:

Environment variables:

OMERO_USERDIR Set the base directory containing the user’s files.
Default: $HOME/omero
OMERO_SESSIONDIR Set the base directory containing local sessions.
Default: $OMERO_USERDIR/sessions
OMERO_TMPDIR Set the base directory containing temporary files.
Default: $OMERO_USERDIR/tmp

Optional session arguments:

-C, --create Create a new session regardless of existing ones
-s SERVER, --server SERVER OMERO server hostname
-p PORT, --port PORT OMERO server port
-g GROUP, --group GROUP OMERO server default group
-u USER, --user USER OMERO username
-w PASSWORD, --password PASSWORD OMERO password
-k KEY, --key KEY OMERO session key (UUID of an active session)
--sudo ADMINUSER Create session as this admin. Changes meaning of password!
-q, --quiet Quiet mode. Causes most warning and diagnostic messages to be suppressed

Group creation

New groups can be added to the database using the omero group add command:

$ bin/omero group add -h

During the addition of the new group, you need to specify the name of the new group:
$ bin/omero group add newgroup

The permissions of the group are set to private by default. Alternatively you can specify the permissions using --perms or --type optional arguments:

$ bin/omero group add read-only-1 --perms="rwr---"
$ bin/omero group add read-annotate-1 --type=read-annotate

See also:

Groups and permissions system Description of the three group permissions levels (private, read-only, read-annotate). Lists of users/groups on the OMERO server can be queried using the omero user list and omero group list commands:

$ bin/omero user list
$ bin/omero group list

Group management

Users can be added to existing groups using the omero user joingroup or omero group adduser commands. Similarly, users can be removed from existing groups using the omero user leavegroup or omero group removeuser commands:

# Add jsmith to group read-annotate-1
$ bin/omero group adduser jsmith --name=read-annotate-1
# Remove jsmith from group read-annotate-1
$ bin/omero group removeuser jsmith --name=read-annotate-1
# Add jsmith to group read-only-1
$ bin/omero user joingroup read-only-1 --name=jsmith
# Remove jsmith from group read-only-1
$ bin/omero user leavegroup read-only-1 --name=jsmith

By passing the --as-owner option, these commands can also be used to manage group owners:

# Add jsmith to the owner list of group read-annotate-1
$ bin/omero group adduser jsmith --name=read-annotate-1 --as-owner
# Remove jsmith from the owner list of group read-annotate-1
$ bin/omero user leavegroup read-annotate-1 --name=jsmith --as-owner

Group copy

To create a copy of a group, you must first create a new group using the omero group add command:

$ bin/omero group add read-only-2 --perms="rwr---"

Then you can use the omero group copyusers command to copy all group members from one group to another:

$ bin/omero group copyusers read-only-1 read-only-2

To copy the group owners, use the same command with the --as-owner optional argument:
$ bin/omero group copyusers read-only-1 read-only-2 --as-owner

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 6.3.5 Repository management

Since 5.0.3 it is possible to list images, filesets and the repositories that contain them. At an administrator-only level it is also possible to move filesets. This functionality is provided by the `omero fs` command. See

$ bin/omero fs -h

#### Listing repositories

The `omero fs repos` subcommand lists the repositories used by OMERO. For example

```
bin/omero fs repos
```

| # | Id | UUID | Type | Path                      |
|---|----|--------------------------------------+---------|--------------------------|
| 0 | 1 | 83bf5c68-8236-47ff-ae3e-728674eb0103 | Managed | /OMERO/ManagedRepository |
| 1 | 2 | ad899754-bff0-4605-a234-acd4da178f3b | Public  | /OMERO                  |
| 2 | 3 | ScriptRepo | Script | /dist/lib/scripts       |

The options available to this subcommand are:

* `-h, --help`
  
  Display the help for this subcommand.

* `--style <{plain, csv, sql}>`
  
  This option determines the output style, tabular `sql` being the default as in the previous example. The `csv` style is comma-separated values with an initial header row, `plain` is the same as `csv` but without the header row.

* `--managed`

  This option lists only Managed repositories.

For example

```
bin/omero fs repos --managed --style=csv
```

#,Id,UUID,Type,Path
0,1,83bf5c68-8236-47ff-ae3e-728674eb0103,Managed,/OMERO/ManagedRepository

#### Listing filesets

The `omero fs sets` subcommand lists filesets by various criteria. Filesets are bundles of original data imported into OMERO 5 and above, which represent one or more images. For example

```
bin/omero fs sets
```

<table>
<thead>
<tr>
<th>#</th>
<th>Id</th>
<th>Prefix</th>
<th>Images</th>
<th>Files</th>
<th>Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>79853</td>
<td>user-3_9/2014-07/22/16-41-04.244</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>79852</td>
<td>user-3_9/2014-07/22/10-44-11.235</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
The options available to this subcommand are:

-h, --help
Display the help for this subcommand.

--style <{plain,csv,sql}>
See omero fs repos --style.

--limit <LIMIT>
This option specifies the maximum number of return values, the default is 25.

--offset <OFFSET>
This option specifies the number of entries to skip before starting the listing, the default, 0, is to skip no entries.

--order <{newest, oldest, prefix}>
This option determines the order of the rows returned, newest is the default.

--without-images
This option lists only those filesets without images, these may be corrupt filesets.

--with-transfer <WITH_TRANSFER [WITH_TRANSFER ...]>
This option lists only those filesets imported using the given in-place import methods.

--check
This option checks each fileset for validity by recalculating each file’s checksum and comparing it with the checksum recorded upon import. This may be slow. This option is available to administrators only.

--extended
With this option more details are provided for each returned value. This may be slow.

For example

```
bin/omero fs sets --order oldest --limit 3 --offset 5 --check
```

<table>
<thead>
<tr>
<th>#</th>
<th>Id</th>
<th>Prefix</th>
<th>Images</th>
<th>Files</th>
<th>Transfer</th>
<th>Check</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>54</td>
<td>user-3_9/2014-06/09/09-24-28.037/</td>
<td>1</td>
<td>1</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>1</td>
<td>55</td>
<td>user-3_9/2014-06/09/09-24-31.354/</td>
<td>1</td>
<td>1</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>2</td>
<td>57</td>
<td>user-5_4/2014-06/09/11-01-00.557/</td>
<td>1</td>
<td>1</td>
<td></td>
<td>OK</td>
</tr>
</tbody>
</table>

(3 rows, starting at 5 of approx. 78415)

Listing images

The omero fs images subcommand lists imported images by various criteria. This subcommand is useful for showing pre-FS (i.e. OMERO 4.4 and before) images which have their original data archived with them. For example

```
bin/omero fs images
```

<table>
<thead>
<tr>
<th>#</th>
<th>Image</th>
<th>Name</th>
<th>FS</th>
<th># Files</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>102803</td>
<td>kidney_TF1_l.bmp.ome.tiff</td>
<td>79853</td>
<td>1</td>
<td>435.1 KB</td>
</tr>
<tr>
<td>1</td>
<td>102802</td>
<td>4kx4k.jpg</td>
<td>79852</td>
<td>1</td>
<td>1.7 MB</td>
</tr>
</tbody>
</table>

6.3. Command Line Interface as an OMERO admin tool 181
The options available to this subcommand are:

- **-h, --help**
  Display the help for this subcommand.

- **--style** {plain, csv, sql}
  See `omero fs repos --style`.

- **--limit** <LIMIT>
  See `omero fs sets --limit`.

- **--offset** <OFFSET>
  See `omero fs sets --offset`.

- **--order** {newest, oldest, prefix}
  See `omero fs sets --order`.

- **--archived**
  With this option the subcommand lists only images with archived data.

- **--extended**
  With this option more details are provided for each returned value. This may be slow.

For example

```
bin/omero fs images --archived --offset 16 --limit 3
```

```
# | Image | Name | FS   | # Files | Size
---+-------+-------+-------+---------+---------
0  | 15481 | UMD001_ORO.svs [Series 1] | | 1       | 12.7 MB
1  | 15478 | biosamplefullframetif.tif | | 1       | 32.0 MB
2  | 10018 | 050118.lei [07-13-a]     | | 4       | 4.8 MB
```

(3 rows, starting at 16 of approx. 833)

**Renaming filesets**

The `omero fs rename` subcommand moves an existing fileset, specified by its ID, to a new location. **This subcommand is available to administrators only.**

It may be useful to rename an existing fileset after the import template (`omero.fs.repo.path`) has been changed to match the new template. By default the files in the fileset and the accompanying import log are moved. For example, after adding the group name and group ID to template and changing the date format

```
$ bin/omero fs rename 9
```

Renaming Fileset:9 to pg-0_3/user-0_2/2014-07-23/16-48-20.786/

The ID can be given as a number or in the form `Fileset:ID`.

The options available to this subcommand are:
-h, --help
Display the help for this subcommand.

--no-move
With this option the files will be left in place to be moved later. Advice will be given as to which files need to be moved to complete the renaming process. Note that if the files are not moved then the renamed filesets will not be accessible until the files have been moved into the new positions.

For example

```
$ bin/omero fs rename Fileset:8 --no-move
Renaming Fileset:8 to pg-0_3/user-0_2/2014-07-23/16-49-23.543/
Done. You will now need to move these files manually:
-----------------------------------------------------
```

Detailing disk usage

The `omero fs usage` subcommand provides details of the underlying disk usage for various types of objects. This subcommand takes optional positional arguments of object types with ids and returns the total disk usage of the specified objects.

For example

```
bin/omero fs usage Image:30001,30051 Plate:1051 --report
```

```
Total disk usage: 1064320138 bytes in 436 files
component       | size (bytes) | files
-----------------+--------------+-------
Thumbnail        | 582030       | 256   
Job              | 1772525      | 2     
Pixels           | 49545216     | 12    
FilesetEntry     | 1011947729   | 124   
Annotation       | 472638       | 42    
```

If no positional argument is given then the total usage for the current user across all of that user’s groups is returned.

For example

```
bin/omero fs usage --report
```

```
Total disk usage: 4526436430274 bytes in 26078 files
component       | size (bytes) | files
-----------------+---------------+-------
Pixels           | 14654902013   | 2961  
FilesetEntry     | 4510839804505 | 8820  
Thumbnail        | 17337131      | 8110  
Job              | 265665153      | 2792  
OriginalFile     | 1757277        | 109   
Annotation       | 13167582976    | 3910  
```

If multiple objects are given and those objects contain common data then that usage will not be counted twice. For example, if two datasets contain the same image then the fileset for that image will not be double-counted in the total disk usage.

The options available to this subcommand are:

- **-h, --help**: Display the help for this subcommand.
--style {plain, csv, sql}
    See omero fs repos --style.

--wait WAIT
    Number of seconds to wait for the processing to complete. To wait indefinitely use < 0, for no wait use 0. The default is to wait indefinitely.

--size_only
    Print total bytes used, in bytes, with no extra text, this is useful for automated scripting.

--report
    Print detailed breakdown of disk usage by types of files. This option is ignored if --size_only is used.

--units {K, M, G, T, P}
    Units to use for disk usage for the total size using base-2. The default is bytes.

--groups
    Print size for all of the current user’s groups, this includes the user’s own data and the data of other group members visible to the user. This option only applies if no positional arguments are given.

For example

bin/omero fs usage --groups --size_only -C -u user-1
4576108188820

bin/omero fs usage Project:1,2 Dataset:5 --units M --report

    Total disk usage: 1432 MiB in 121 files
    component | size (bytes) | files
    --------------------------
    Thumbnail      73710     34
    Pixels        1499341282  34
    Annotation    3000028     53
    (3 rows)

See also:

Command Line Interface as an OMERO client   User documentation for the Command Line Interface
Command Line Interface as an OMERO development tool   Developer Documentation for the Command Line Interface

Help for any specific CLI command can be displayed using the -h argument. See Command line help for more information.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.4 Upgrading and maintenance

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.4.1 OMERO.server backup and restore

Cleaning up your binary repository

The OMERO.server does not remove files from disk until a cleanup task has been run. A script to do this is included in the OMERO.server distribution lib/python/omero/util/cleanse.py which can be used so:

6.4. Upgrading and maintenance 184
$ bin/omero admin cleanse /OMERO

This can be performed daily using cron with a script such as:

```sh
#!/bin/bash
USERNAME="root"
PASSWORD="root_password"
BINARY_REPOSITORY="/OMERO"
$OMERO_PREFIX/bin/omero -s localhost -u $USERNAME -w $PASSWORD admin cleanse $BINARY_REPOSITORY
```

**Warning:** If you are running OMERO 5.2.3 or earlier, do not run this script as an operating system user while logged into OMERO as a non-administrative user as this will lead to data loss. Instead, you should always run as an administrative user such as “root”. See this [announcement](http://www.openmicroscopy.org/community/viewtopic.php?f=11&t=8060) for further details. More recent versions enforce this by making the script and command admin-only.

---

### Managing OMERO.server log files

Your OMERO.server will produce log files that are rotated when they reach 512MB. These directories will look like:

```
omero_dist $ ls var/log
Blitz-0.log FileServer.log MonitorServer.log Processor-0.log master.out
DropBox.log Indexer-0.log OMEROweb.log master.err
```

Any files with a .1, .2, .3 etc. suffix may be compressed or deleted.

### OMERO.server log file location

The log file directory may also be relocated to different storage by modifying the `etc/grid/default.xml` file:

```
...<variable name="OMERO_LOGS" value="var/log"/>
...```

---

### Backing up OMERO

#### Understanding backup sources

OMERO.server has three main backup sources:

1. PostgreSQL database (assumed to be `omero_database`)
2. OMERO.server binary data store ([UNIX/Mac information page](https://www.openmicroscopy.org.uk/omero/query/server.html#Installation) or [Windows information page](https); assumed to be `/OMERO` or `C:\OMERO`)
3. OMERO.server configuration

**Note:** The lib/scripts directory should also be backed up, but restoring it may pose issues if any of your users have added their own “official scripts”. A github repository is now available under [https://github.com/ome/scripts](https://github.com/ome/scripts) which provides help for merging your lib/scripts directories.

6.4. Upgrading and maintenance 185
You should back up (1) and (2) regularly.

**Warning:** Although losing your PostgreSQL database is not as catastrophic for OMERO version 5.0 as it is for 4.4, you still do not want to be in the position of trying to recover your server without a backup of this resource.

You need to back up (3) only before you make changes. You can copy it into /OMERO/backup to ensure it is kept safe:

```
$ bin/omero config get > /OMERO/backup/omero.config
```

**Note:** If you have edited `etc/grid/(win)default.xml` directly for any reason then you will also need to copy that file to somewhere safe, such as /OMERO/backup.

**Back up your PostgreSQL database**

Database backups can be achieved using the PostgreSQL `pg_dump` command. Here is an example backup script that can be placed in `/etc/cron.daily` to perform daily database backups:

```
#!/bin/bash

DATE='date '+%Y-%m-%d_%H:%M:%S' -z''
OUTPUT_DIRECTORY=/OMERO/backup/database
DATABASE="omero database"
DATABASE_ADMIN="postgres"

mkdir -p $OUTPUT_DIRECTORY
chown -R $DATABASE_ADMIN $OUTPUT_DIRECTORY
su $DATABASE_ADMIN -c "pg_dump -Fc -f $OUTPUT_DIRECTORY/$DATABASE.$DATE.pg_dump $DATABASE"
```

Other database backup configurations are outside the scope of this document but can be researched on the PostgreSQL website\(^\text{161}\) (Chapter 24. Backup and Restore).

**Note:** Regular backups of your PostgreSQL database are crucial; you do not want to be in the position of trying to restore your server without one.

**Back up your binary data store**

To simplify backup locations we have, in this document, located all database and configuration backups under /OMERO, your binary data store. The entire contents of /OMERO should be backed up regularly as this will, especially if this document’s conventions are followed, contain all the relevant data to restore your OMERO.server installation in the unlikely event of a system failure, botched upgrade or user malice.

File system backup is often a very personal and controversial topic amongst systems administrators and as such the OMERO project does not make any explicit recommendations about backup software. In the interest of providing a working example we will use open source rdiff-backup project and like **Back up your PostgreSQL database** above, provide a backup script which can be placed in `/etc/cron.daily` to perform daily /OMERO backups:

```
#!/bin/bash

FROM=/OMERO
TO=/mnt/backup_server

rdiff-backup $FROM $TO
```

\(^\text{161}\)http://www.postgresql.org/docs/9.4/interactive/backup.html

6.4. Upgrading and maintenance
rdiff-backup can also be used to backup /OMERO to a remote machine:

```bash
#!/bin/bash
FROM=/OMERO
TO=backup_server.example.com::/backup/omero
rdiff-backup $FROM $TO
```

More advanced rdiff-backup configurations are beyond the scope of this document. If you want to know more you are encouraged to read the documentation available on the rdiff-backup website\(^{162}\).

**Restoring OMERO**

There are three main steps to OMERO.server restoration in the event of a system failure:

1. OMERO.server etc configuration
2. PostgreSQL database (assumed to be omero)
3. OMERO.server binary data store (assumed to be /OMERO)

**Note:** It is important that restoration steps are done in this order unless you are absolutely sure what you are doing.

**Restoring your configuration**

Once you have retrieved an OMERO.server package from the downloads\(^{163}\) page that matches the version you originally had installed, all that is required is to restore your backup preferences by running:

```
$ bin/omero config load /OMERO/backup/omero.config
```

You should then follow the Reconfiguration steps of install.

**Restoring your PostgreSQL database**

If you have had a PostgreSQL crash and database users are missing from your configuration, you should follow the first two (Create a non-superuser database user and Create a database for OMERO data to reside in) steps of OMERO.server installation. Once you have ensured that the database user and empty database exist, you can restore the pg_dump file as follows:

```
$ sudo -u postgres pg_restore -Fc -d omero_database omero.2010-06-05_16:27:29-GMT.pg_dump
```

**Restoring your OMERO.server binary data store**

All that remains once you have restored your Java preferences and PostgreSQL database is to restore your /OMERO binary data store backup.

**See also:**

- List of backup software\(^{164}\) Wikipedia page listing the backup softwares.
- PostgreSQL 9.4 Interactive Manual\(^{165}\) Chapter 24: Backup and Restore
- rdiff-backup documentation\(^{166}\) Online documentation of rdiff-backup project

\(^{162}\)http://www.nongnu.org/rdiff-backup/docs.html
\(^{163}\)http://downloads.openmicroscopy.org/latest/omero/
\(^{164}\)http://www.nongnu.org/rdiff-backup/docs.html
\(^{165}\)http://downloads.openmicroscopy.org/latest/omero/
Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.4.2 OMERO.server upgrade

The OME team is committed to providing frequent, project-wide upgrades both with bug fixes and new functionality. We try to make the schedule for these releases as public as possible. You may want to take a look at the roadmap167 for exactly what will go into a release. We always inform our mailing lists168 of the development status.

See the full details of OMERO 5.2.6 features in the Announcements169 forum.

This guide aims to be as definitive as possible so please do not be put off by the level of detail; upgrading should be a straightforward process.

Upgrade check list

- Check prerequisites
- File limits
- Password usage
- OMERO.web configuration
- OMERO.web dependencies
- Web plugin updates
- Memoization files invalidation
- Troubleshooting
- Upgrade check

Check prerequisites

Before starting the upgrade, please ensure that you have reviewed and satisfied all the system requirements with correct versions for installation. In particular, ensure that you are running a suitable version of PostgreSQL to enable successful upgrading of the database, otherwise the upgrade script aborts with a message saying that your database server version is less than the OMERO prerequisite. If you are upgrading from a version earlier than OMERO 5.0 then first review the 5.0 upgrade notes170 regarding previous changes in OMERO.

File limits

You may wish to review the open file limits. Please consult the Too many open file descriptors section for further details.

Password usage

The passwords and logins used here are examples. Please consult the Which user account and password do I use where? section for explanation. In particular, be sure to replace the values of db_user and omero_database with the actual database user and database name for your installation.

OMERO.web configuration

Check Deployment (Unix/Linux) or Deployment (Windows), what is the most recent OMERO.web deployment for the platform. If you generated configuration stanzas using omero web config which enables OMERO.web via Apache or Nginx, they will include hard-coded links to your previous version of OMERO. Therefore, you should regenerate your config files when

---

167https://trac.openmicroscopy.org/ome/roadmap
168http://www.openmicroscopy.org/site/community/
170http://www.openmicroscopy.org/site/support/omero5.0/sysadmins/server-upgrade.html
upgrading, remembering to merge in any of your own modifications if necessary. You should carry out this step even for minor version upgrades as there may be fixes which require it.

**Note:** Since OMERO 5.2, the OMERO.web framework no longer bundles a copy of the Django package, instead manual installation of the Django dependency is required. It is highly recommended to use Django 1.8\(^{171}\) (LTS) which requires Python 2.7. For more information see Python on the Version requirements page.

Also note that as of OMERO 5.2.6, support for Apache deployment is deprecated and is likely be dropped during the 5.3.x line.

### OMERO.web dependencies

While upgrading the server it is recommended to keep OMERO.web dependencies up to date to ensure that security updates are applied.

- **Nginx on Unix:**

  
  $ pip install --upgrade -r share/web/requirements-py27-nginx.txt

- **Apache (deprecated) on Unix:**

  
  $ pip install --upgrade -r share/web/requirements-py27-apache.txt

- **Windows:**

  
  $ pip install --upgrade -r share/web/requirements-py27-win.txt

**Note:** This is mainly for new major releases. If carrying out a minor release upgrade, you should check that updates will not break your set-up before performing this on your production system.

### Web plugin updates

OMERO.web plugins are very closely integrated into the webclient. For this reason, it is possible that an update of OMERO will cause issues with an older version of a plugin. It is best when updating the server to also install any available plugin updates according to their own documentation.

### Memoization files invalidation

All cached Bio-Formats memoization files created at import time will be invalidated by the server upgrade. This means the very first loading of each image after upgrade will be slower. After re-initialization, a new memoization file will be automatically generated and OMERO will be able to load images in a performant manner again.

These files are stored under `BioFormatsCache` in the OMERO data directory, e.g. `/OMERO/BioFormatsCache`. You may see error messages in your log files when an old memoization file is found; to avoid these messages delete everything under this directory before starting the upgraded server.

### Troubleshooting

If you encounter errors during an OMERO upgrade, database upgrade, etc. you should retain as much log information as possible and notify the OMERO.server team via the mailing lists available on the community\(^{172}\) page.

\(^{171}\)https://docs.djangoproject.com/en/1.8/releases/1.8/

\(^{172}\)http://www.openmicroscopy.org/site/community/
Upgrade check

All OMERO products check themselves with the OmeroRegistry for update notifications on startup. If you wish to disable this functionality you should do so now as outlined on the OMERO upgrade checks page.

Upgrade steps

For all users, the basic workflow for upgrading your OMERO.server is listed below. Please refer to each section for additional details.

- Perform a database backup
- Copy new binaries
- Upgrade your database
- Merge script changes
- Update your environment variables and memory settings
- Update your OMERO.web server configuration
- Restart your server
- Restore a database backup

Perform a database backup

The first thing to do before any upgrade activity is to backup your database.

```bash
$ pg_dump -h localhost -U db_user -Fc -f before_upgrade.db.dump omero_database
```

Copy new binaries

Before copying the new binaries, stop the existing server:

```bash
$ cd OMERO.server
$ bin/omero web stop
$ bin/omero admin stop
```

Your OMERO configuration is stored using config.xml in the etc/grid directory under your OMERO.server directory. Assuming you have not made any file changes within your OMERO.server distribution directory, you are safe to follow the following upgrade procedure:

```bash
$ cd ..
$ mv OMERO.server OMERO.server-old
$ unzip OMERO.server-5.2.6-ice3x-byy.zip
$ ln -s OMERO.server-5.2.6-ice3x-byy OMERO.server
$ cp OMERO.server-old/etc/grid/config.xml OMERO.server/etc/grid
```

Note: ice3x and byy need to be replaced by the appropriate Ice version and build number of OMERO.server.

Upgrade your database

**Warning:** This section only concerns users upgrading from a 5.1 or earlier server. If upgrading from a 5.2 server, you do not need to upgrade the database.
Ensure Unicode character encoding  Versions of OMERO from 5.1.0 onwards require a Unicode-encoded database; without it, the upgrade script aborts with a message warning how the “OMERO database character encoding must be UTF8”. From *psql*:

```
# SELECT datname, pg_encoding_to_char(encoding) FROM pg_database;
 datname | pg_encoding_to_char  
------------+---------------------
 template1 | UTF8
 template0 | UTF8
 postgres | UTF8
 omero | UTF8
 (4 rows)
```

Alternatively, simply run *psql* -l and check the output. If your OMERO database is not Unicode-encoded with UTF8 then it must be re-encoded.

If you have the `pg_upgradecluster` command available then its `--locale` option can effect the change in encoding. Otherwise, create a Unicode-encoded dump of your database: dump it *as before* but to a different dump file and with an additional `--E UTF8` option. Then, create a Unicode-encoded database for OMERO and restore that into it with *pg_restore*, similarly to *[effecting a rollback]*. If required to achieve this, the `--E UTF8` option is accepted by both *initdb* and *createdb*.

**Run the upgrade script**  You **must** use the same username and password you have defined during *OMERO.server installation*. For a large production system you should plan for the fact that the upgrade script may take several hours to run.

```
$ cd OMERO.server
$ psql -h localhost -U db_user omero_database < sql/psql/OMERO5.2__0/OMERO5.1__1.sql
Password for user db_user:
...
...
status

YOU HAVE SUCCESSFULLY UPGRADED YOUR DATABASE TO VERSION OMERO5.2__0
```

(1 row)

If you are upgrading from a server earlier than 5.1 then it suffices to run the earlier upgrade scripts in sequence before the one above. There is no need to download and run the server from an intermediate major release.

**Note:** If you perform the database upgrade using *SQL shell*, make sure you are connected to the database using *db_user* before running the script. See [this forum thread](http://www.openmicroscopy.org/community/viewtopic.php?f=5&t=7778) for more information.

**Optimize an upgraded database (optional)**  After you have run the upgrade script, you may want to optimize your database which can both save disk space and speed up access times.

```
$ psql -h localhost -U db_user omero_database -c 'REINDEX DATABASE 'omerodatabase'' FORCE;
$ psql -h localhost -U db_user omero_database -c 'VACUUM FULL VERBOSE ANALYZE;'
```

**Patch database functions (optional)**  An additional database script is provided for fixing an error that affects only systems that were once upgraded from OMERO 4.4. More recent OMERO installations should skip this patching step.

The script patches OMERO 5.1 and 5.2 databases, runs quickly, and can be applied anytime, even under a running OMERO server. Although the problem has been sufficiently asymptomatic to go unnoticed for some time we nonetheless recommend that the patch is applied promptly.

---

$ psql -h localhost -U db_user omero_database < sql/psql/OMERO5.2__0/OMERO5.2-5.1-format-syntax-patch.sql

An unpatched system may exhibit serious problems if run with PostgreSQL 9.5. OMERO 5.2 does not support PostgreSQL 9.5 and the upgrade script for OMERO 5.3 will ensure that these PL/pgSQL functions are patched regardless of whether you used this provided script before that upgrade.

**Merge script changes**

If any new official scripts have been added under `lib/scripts` or if you have modified any of the existing ones, then you will need to backup your modifications. Doing this, however, is not as simple as copying the directory over since the core developers will have also improved these scripts. In order to facilitate saving your work, we have turned the scripts into a Git submodule which can be found at [https://github.com/ome/scripts](https://github.com/ome/scripts).

For further information on managing your scripts, refer to [OMERO.scripts](https://github.com/ome/scripts). If you require help, please contact the OME developers.

**Update your environment variables and memory settings**

**Environment variables** If you changed the directory name where the 5.2.6 server code resides, make sure to update any system environment variables. Before restarting the server, make sure your `PATH` and `PYTHONPATH` system environment variables are pointing to the new locations.

**JVM memory settings** Your memory settings should be copied along with `etc/grid/config.xml`, but you can check the current settings by running `omero admin jvmcfg`. See [Memory configuration](https://github.com/ome/scripts) for more information.

**Update your OMERO.web server configuration**

FastCGI support was removed in OMERO 5.2 and OMERO.web can be deployed using WSGI (see [Deployment (Unix/Linux)](https://github.com/ome/scripts) for more details). If you have already deployed OMERO.web using WSGI you should regenerate your config files, remembering to merge in any of your own modifications if necessary. Due to the nature of OMERO.web development for the 5.2.x line, you should carry out this step even for minor version upgrades as there may be fixes which require it.

If necessary ensure you have set up a regular task to clear out any stale OMERO.web session files as described in [OMERO.web Maintenance (Unix/Linux)](https://github.com/ome/scripts) or [OMERO.web Maintenance (Windows)](https://github.com/ome/scripts).

**Restart your server**

- Following a successful database upgrade, you can start the server.

  $ cd OMERO.server
  $ bin/omero admin start

- If anything goes wrong, please send the output of `omero admin diagnostics` to `ome-users@lists.openmicroscopy.org.uk`.

- Start OMERO.web with the following command:

  $ bin/omero.web start

**Restore a database backup**

If the upgraded database or the new server version do not work for you, or you otherwise need to rollback to a previous database backup, you may want to restore a database backup. To do so, create a new database,
$ createdb -h localhost -U postgres -E UTF8 -O db_user omero_from_backup
restore the previous archive into this new database,

$ pg_restore -Fc -d omero_from_backup before_upgrade.db.dump

and configure your server to use it,

$ bin/omero config set omero.db.name omero_from_backup

See also:
Legacy\textsuperscript{175} Legacy part of the OME website containing upgrade instructions for previous versions of the OMERO server.

\textbf{Note:} This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

\section*{6.4.3 OMERO upgrade checks}

On each startup the OMERO server checks for available upgrades via the UpgradeCheck class\textsuperscript{176}. An HTTP GET call is made to the URL configured in etc/omero.properties as omero.upgrades.url, currently http://upgrade.openmicroscopy.org.uk by default (note that viewing that link in your browser will redirect you to this page).

\textbf{Note:} If you have been redirected here by clicking on a link to http://upgrade.openmicroscopy.org.uk in an error message or log while trying to run one of the Bio-Formats command line tools, please see the Bio-Formats command line tools documentation\textsuperscript{178} for assistance.

\section*{Actions}

Currently the only action taken when an upgrade is necessary is a log statement at WARN level.


Future versions may also send emails and/or IMs to administrators. In the case of critical upgrades, the server may refuse to start.

\section*{Privacy}

Currently, the only information which is being transmitted to the server is:

\begin{itemize}
\item Java virtual machine version
\item operating system details (architecture, version and name)
\item current server version
\item poll frequency (for determining statistics)
\item your IP address (standard HTTP header information)
\end{itemize}

\textbf{Note:} Currently the “poll” property is unused.

\textsuperscript{175}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/system/UpgradeCheck.java
\textsuperscript{176}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/etc/omero.properties
\textsuperscript{178}http://www.openmicroscopy.org/site/support/bio-formats/users/comlinetools/index.html#version-checker
Disabling

If you would prefer to have no checks made, the check can be disabled by setting the `omero.upgrades.url` property to an empty string:

`omero.upgrades.url=`

Developers

To use the `UpgradeCheck` class from your own code, it is necessary to have `common.jar` on your classpath. Then,

```java
ResourceBundle bundle = ResourceBundle.getBundle("omero");
String version = bundle.getString("omero.version");
String url = bundle.getString("omero.upgrades.url");
ome.system.UpgradeCheck check = new UpgradeCheck(
    url, version, "insight"); // Or "importer", etc.
check.run();
check.isUpgradeNeeded();
// optionally
check.isExceptionThrown();
```

will connect to the server and check your current version against the latest release.

See also:

- **OMERO.server installation** Instructions for installing OMERO.server on UNIX & UNIX-like platforms
- **OMERO.server installation** Instructions for installing OMERO.server on Windows platforms
- **OMERO.server upgrade** Instructions for upgrading OMERO.server
- **Server security and firewalls** Description of OMERO security practices

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 6.4.4 Moving the data repository

It may be necessary to move either the whole OMERO data directory or only the Managed Repository to a new location on the file system. This should be done with care following the steps below.

Warning: Before moving OMERO data it is wise to ensure that both the data and the database are fully backed up. See **OMERO.server backup and restore**.

The current location of the data repositories can be found using the `omero fs repos` command:

```
$ bin/omero fs repos
```

```
# | Id | UUID                        | Type          | Path
---+----+----------------------------+----------------+-------------------------------------------------------------
 0 | 1  | 309ea513-a23c-48d1-abd2-9ceed1b3Ha4 | Managed       | /Users/omero/var/omero/ManagedRepository
 1 | 2  | ScriptRepo                 | Script        | /Users/omero/dist/lib/scripts
 2 | 3  | 3ec8c87e-c776-48a3-b57e-2a11b0c97045 | Public        | /Users/omero/var/omero
(3 rows)
```

Note: This command can be slow, `omero config get` can also be used to determine if `omero.data.dir` or `omero.managed.dir` have non-default values.
**Moving the OMERO data directory**

If the Managed Repository is within the OMERO data directory and the whole data directory is to be moved then the following steps should be used:

```
bin/omero admin stop  
bin/omero config set omero.data.dir NEW  
mv OLD NEW  
bin/omero admin start
```

**Warning:** The use of `omero config set` is absolutely necessary here. The steps: `omero admin stop, mv, omero admin start` without `omero config set` could lead to an unstable situation.

For example, moving the OMERO data directory from `/Users/omero/var/omero` to `/Volumes/omero`:

```
$ bin/omero admin stop  
...  
$ bin/omero config set omero.data.dir /Volumes/omero  
$ mv /Users/omero/var/omero /Volumes/omero  
$ bin/omero admin start  
...  
$ bin/omero fs repos
```

<table>
<thead>
<tr>
<th>#</th>
<th>Id</th>
<th>UUID</th>
<th>Type</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>309ea513-a23c-48d1-abd2-9ceed1b3ffa4</td>
<td>Managed</td>
<td>/Volumes/omero/ManagedRepository</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>ScriptRepo</td>
<td>Script</td>
<td>/Users/omero/dist/lib/scripts</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3ee8c878-c77e-48a3-b57e-2a11b0c97045</td>
<td>Public</td>
<td>/Volumes/omero</td>
</tr>
</tbody>
</table>

(3 rows)

**Moving the Managed Repository**

If the Managed Repository is in a separate location from the OMERO data directory or only the Managed Repository is to be moved then the following steps should be used:

```
bin/omero admin stop  
bin/omero config set omero.managed.dir NEW  
mv OLD NEW  
bin/omero admin start
```

**Warning:** The use of `omero config set` is absolutely necessary here. The steps: `omero admin stop, mv, omero admin start` without `omero config set` could lead to an unstable situation.

For example, moving the Managed Repository from `/Users/omero/var/omero/ManagedRepository` to `/Volumes/imports/ManagedRepository`:

```
$ bin/omero admin stop  
...  
$ bin/omero config set omero.managed.dir /Volumes/imports/ManagedRepository  
$ mv /Users/omero/var/omero/ManagedRepository /Volumes/imports/ManagedRepository  
$ bin/omero admin start  
...  
$ bin/omero fs repos
```

<table>
<thead>
<tr>
<th>#</th>
<th>Id</th>
<th>UUID</th>
<th>Type</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>309ea513-a23c-48d1-abd2-9ceed1b3ffa4</td>
<td>Managed</td>
<td>/Volumes/imports/ManagedRepository</td>
</tr>
</tbody>
</table>
6.5 Installing additional features

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.5.1 OMERO.grid

To unify the various components of OMERO, OMERO.grid was developed to monitor and control processes over numerous remote systems. Based on ZeroC’s IceGrid framework, OMERO.grid provides management access, distributed background processing, log handling and several other features.

Terminology

Please notice that ZeroC uses a specific naming scheme for IceGrid elements and actors. A server in the context of this document is not a host computer - it is a process running inside an IceGrid node, servicing incoming requests. A host is a computer on which IceGrid elements get deployed. For more details, see Terminology\(^{180}\).

Getting started

Requirements

The normal OMERO installation actually makes use of OMERO.grid internally. Note that multi-node OMERO.grid configurations are only tested with Unix-like systems, multi-node configurations on Windows are untested and deprecated, and any remaining support will be dropped in OMERO 5.3. If you have followed the instructions under OMERO.server installation you will have everything you need to start working with OMERO.grid.

The standard install should also be used to install other hosts in the grid, such as a computation-only host. Some elements can be omitted for a computation-only host such as PostgreSQL, Apache/nginx, etc.

Running OMERO.web and/or starting up the full OMERO.server instance is not required in such a case (only the basic requirements to run `omero node` are needed, i.e. ZeroC Ice and Python modules for OMERO scripts).

\(^{179}\)https://zeroc.com

\(^{180}\)https://doc.zeroc.com/display/Ice/Terminology
IceGrid Tools

If you would like to explore your IceGrid configuration, use

```
bin/omero admin ice
```

It provides full access to the `icegridadmin` console described in the ZeroC manual. Specific commands can also be executed:

```
bin/omero admin ice help
bin/omero admin ice application list
bin/omero admin ice application describe OMERO
bin/omero admin ice server list
```

Further, by running `java -jar ice-gridgui.jar` the GUI provided by ZeroC can be used to administer OMERO.grid. This JAR is provided in the OMERO source code under `lib/repository`.

See also:
- `icegridadmin Command Line Tool`\(^{181}\) Chapter of the ZeroC\(^{182}\) manual about the `icegridadmin` CLI
- `IceGrid Admin Graphical Tool`\(^{183}\) Chapter of the ZeroC\(^{184}\) manual about the GridGUI tool

OMERO.grid on Windows

Unlike all other supported platforms, the `bin\omero` script and OMERO.grid are not directly responsible for starting and stopping the `OMERO.blitz` server and other processes. Instead, that job is delegated to the native Windows service system. A brief explanation is available under `OMERO.server Windows Service`.

Multi-node deployment is not supported for Windows.

How it works

IceGrid\(^{185}\) is a location and activation service, which functions as a central registry to manage all your OMERO server processes. OMERO.grid provides server components which use the registry to communicate with one another. Other than a minimal amount of configuration and starting a single daemon on each host machine, OMERO.grid manages the complexity of all your computing resources.

Deployment descriptors

All the resources for a single OMERO site are described by one **application descriptor**. OMERO ships with several example descriptors under `etc/grid`. These descriptors describe what processes will be started on what nodes, identified by simple names. For example the default descriptor\(^{186}\), used if no other file is specified, defines the master node. As you will see, these files are critical both for the correct functioning of your server as well as its security.

The deployment descriptors provided define which server instances are started on which nodes. The default descriptor configures the master node to start the `OMERO.blitz` server, the Glacier2 router for firewalling, as well as a single processor - Processor0. The master node is also configured via `etc/master.cfg` to host the registry, though this process can be started elsewhere.

Deployment commands

The master node must be started first to provide the registry. This is done via the `omero admin start` command which uses the default descriptor:

\(^{181}\)https://zeroc.com
\(^{182}\)https://zeroc.com
\(^{183}\)https://doc.zeroc.com/display/Ice/IceGrid
\(^{184}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/etc/templates/grid/default.xml
bin/omero admin start

The deploy command looks for any changes to the defined descriptor and restarts only those servers which have modifications:

bin/omero admin deploy

Both omero admin start and omero admin deploy can optionally take a path to an application descriptor which must be passed on every invocation:

bin/omero admin deploy etc/grid/my-site.xml

Two other nodes, then, each provide a single processor, Processor1 and Processor2. These are started via:

To start a node identified by NAME, the following command can be used

bin/omero node start NAME

At this point the node will try and connect to the registry to announce its presence. If a node with the same name is already started, then registration will fail, which is important to prevent unauthorized users.

The configuration of your grid, however, is very much up to you. Based on the example descriptor files (*.xml) and configuration files (*.cfg), it is possible to develop OMERO.grid installations completely tailored to your computing resources.

The whole grid can be shutdown by stopping the master node via: omero admin stop. Each individual node can also be shutdown via: omero node NAME stop on that particular node.

Deployment examples

Two examples will be presented showing the flexibility of OMERO.grid deployment and identifying files whose modification is critical for the deployment to work.

Nodes on a single host

The first example will focus on changing the deployed nodes/servers on a single host. It should serve as an introduction to the concepts. Unless used for very specific requirements, this type of deployment doesn’t yield any performance gains.

The first change that you will want to make to your application descriptor is to add additional processors. Take a look at etc/templates/grid/default.xml. There you can define two new nodes - node1 and node2 by simply adding a new XML element below the master node definition:

```xml
<node name="node1">
  <server-instance template="ProcessorTemplate" index="1"/>
</node>

<node name="node2">
  <server-instance template="ProcessorTemplate" index="2"/>
</node>
```

Remember to change the node name and the index number for each subsequent node definition. The node name and the index number do not need to match. In fact, the index number can be completely ignored, except for the fact that it must be unique. The node name, however, is important for properly starting your new processor.

You will need both a configuration file under etc/ with the same name, and unless the node name matches the name of your local host, you will need to specify it on the command line:

187https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/etc/templates/grid/default.xml

6.5. Installing additional features
bin/omero node node1 start

or with the environment variable OMERO_NODE:

OMERO_NODE=node1 bin/omero node start

After starting up both nodes, you can verify that you now have three processors running by looking at the output of omero admin diagnostics.

For more information on using scripts, see the OMERO.scripts advanced topics.

Nodes on multiple hosts

**Warning**: Before attempting this type of deployment, make sure that the hosts can ping each other and that required ports are open and not firewalled.

Windows multi-node configurations are not supported.

A more complex deployment example is running multiple nodes on networked hosts. Initially, the host's loopback IP address (127.0.0.1) is used in the grid configuration files.

For this example, let's presume we have control over two hosts: omero-master (IP address 192.168.0.1/24) and omero-slave (IP address 192.168.0.2/24). The goal is to move the processor server onto another host (omero-slave) to reduce the load on the host running the master node (omero-master). The configuration changes required to achieve this are outlined below.

On host omero-master:

- etc/grid/default.xml - remove or comment out from the master node the server-instance using the ProcessorTemplate. Below the master node add an XML element defining a new node:

  ```
  <node name="omero-slave">
    <server-instance template="ProcessorTemplate" index="0" dir=""/>
  </node>
  ```

- etc/internal.cfg - change the value of Ice.Default.Locator from 127.0.0.1 to 192.168.0.1

- etc/master.cfg - change all occurrences of 127.0.0.1 to 192.168.0.1

On host omero-slave:

- copy or rename etc/node1.cfg to etc/omero-slave.cfg and change all node1 strings to omero-slave in etc/omero-slave.cfg. Also update the IceGrid.Node.Endpoints value to tcp -h 192.168.0.2

- etc/internal.cfg - change the value of Ice.Default.Locator from 127.0.0.1 to 192.168.0.1

- etc/ice.config - add the line Ice.Default.Router=OMERO.Glacier2/router:tcp -p 4063 -h 192.168.0.1

To apply the changes, start the OMERO instance on the omero-master node by using omero admin start. After that, start the omero-slave node by using omero node omero-slave start. Issuing omero admin diagnostics on the master node should show a running processor instance and the omero-slave node should accept job requests from the master node.

Securing grid resources

More than just making sure no malicious code enters your grid, it is critical to prevent unauthorized access via the application descriptors (*.xml) and configuration (*.cfg) as mentioned above.
Firewall

The simplest and most effective way of preventing unauthorized access is to have all OMERO.grid resources behind a firewall. Only the Glacier2 router has a port visible to machines outside the firewall. If this is possible in your configuration, then you can leave the internal endpoints unsecured.

SSL (Secure Socket Layer)

Though it is probably unnecessary to use transport encryption within a firewall, encryption from clients to the Glacier2 router will often be necessary. For more information on SSL, see SSL.

Permissions Verifier

The IceSSL plugin can be used both for encrypting the channel as well as authenticating users. SSL-based authentication, however, can be difficult to configure especially for within the firewall, and so instead you may want to configure a “permissions verifier” to prevent non-trusted users from accessing a system within your firewall. From master.cfg:\n
\begin{verbatim}
IceGrid.Registry.AdminPermissionsVerifier=IceGrid/NullPermissionsVerifier
#IceGrid.Registry.AdminCryptPasswords=etc/passwd
\end{verbatim}

Here we have defined a “null” permissions verifier which allows anyone to connect to the registry’s administrative endpoints. One simple way of securing these endpoints is to use the AdminCryptPasswords property, which expects a passwd-formatted file at the given relative or absolute path:

\begin{verbatim}
mrmypasswordisomero TN7CjkTVoDnb2
msmypasswordisome jky23t9JXPRRU
\end{verbatim}

where these values come from using openssl:

\begin{verbatim}
$ openssl
OpenSSL> passwd
Password: |
Verifying - Password: |
TN7CjkTVoDnb2
OpenSSL>
\end{verbatim}

Another possibility is to use the OMERO.blitz permissions verifier, so that anyone with a proper OMERO account can access the server.

See Controlling Access to IceGrid Sessions\textsuperscript{189} of the Ice manual for more information.

Unique node names

Only a limited number of node names are configured in an application descriptor. For an unauthorized user to fill a slot, they must know the name (which is discoverable with the right code) and be the first to contact the grid saying “I am Node029”, for example. A system administrator need only then be certain that all the node slots are taken up by trusted machines and users.

It is also possible to allow “dynamic registration” in which servers are added to the registry after the fact. In some situations this may be quite useful, but is disabled by default. Before enabling it, be sure to have secured your endpoints via one of the methods outlined above.

\begin{footnotesize}
188https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/etc/templates/master.cfg
189https://doc.zeroc.com/display/Ice/Resource+Allocation+using+IceGrid+Sessions#ResourceAllocationusingIceGridSessions- ControllingAccessstoIceGridSessions
\end{footnotesize}
Absolute paths

Except under Windows, the example application descriptors shipped with OMERO, all use relative paths to make installation easier. Once you are comfortable with configuring OMERO.grid, it would most likely be safer to configure absolute paths. For example, specifying that nodes execute under /usr/lib/omero requires that whoever starts the node have access to that directory. Therefore, as long as you control the boxes which can attach to your endpoints (see Firewall), then you can be relatively certain that no tampering can occur with the installed binaries.

Technical information and other tips

Processes

It is important to understand just what processes will be running on your servers. When you run omero admin start, icegridnode is executed which starts a controlling daemon and deploys the proper descriptor. This configuration is persisted under var/master and var/registry.

Once the application is loaded, the icegridnode daemon process starts up all the servers which are configured in the descriptor. If one of the processes fails, it will be restarted. If restart fails, eventually the server will be “disabled”. On shutdown, the icegridnode process also shuts down all the server processes.

Targets

In application descriptors, it is possible to surround sections of the description with <target/> elements. For example, in templates.xml the section which defines the main OMERO.blitz server includes:

```xml
<server id="Blitz-${index}" exe="${JAVA}" activation="always" pwd="${OMERO_HOME}">
  <target name="debug">
    <option>-Xdebug</option>
    <option>-Xrunjdwp:server=y,transport=dt_socket,address=8787,suspend=y</option>
  </target>
  ...
```

When the application is deployed, if “debug” is added as a target, then the -Xdebug, etc. options will be passed to the Java runtime. This will allow remote connection to your server over the configured port.

Multiple targets can be enabled at the same time:

```
bino/mero admin deploy etc/grid/default.xml debug secure someothertarget
```

Ice.MessageSizeMax

Ice imposes an upper limit on all method invocations. This limit, Ice.MessageSizeMax, is configured in your application descriptor (e.g. templates.xml) and configuration files (e.g. ice.config). The setting must be applied to all servers which will be handling the invocation. For example, a call to InteractiveProcessor.execute(omero::RMap inputs) which passes the inputs all the way down to processor.py will need to have a sufficiently large Ice.MessageSizeMax for: the client, the Glacier2 router, the OMERO.blitz server, and the Processor.

The default is currently set to 65536 kilobytes which is 64MB.

Logging

Currently all output from OMERO.grid is stored in $OMERO_PREFIX/var/log/master.out with error messages going to $OMERO_PREFIX/var/log/master.err. Individual services may also create their own log files.

6.5. Installing additional features

190 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/etc/templates/grid/templates.xml
191 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/etc/templates/grid/templates.xml
192 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/etc/templates/ice.config
Shortcuts

If the bin/omero script is copied or symlinked to another name, then the script will separate the name on hyphens and execute bin/omero with the second and later parts prepended to the argument list.

For example,

```bash
ln -s bin/omero bin/omero-admin
bin/omero-admin start
```

works identically to:

```bash
bin/omero admin start
```

Symbolic linking

Shortcuts allow the bin/omero script to function as an init.d script when named omero-admin, and need only be copied to /etc/init.d/ to function properly. It will resolve its installation directory, and execute from there unless OMERO_HOME is set.

For example,

```bash
ln -s $OMERO_PREFIX/bin/omero /usr/local/bin/omero
omeroinit start
```

The same works for putting bin/omero on your path:

```
PATH=$OMERO_PREFIX/bin:$PATH
```

This means that OMERO.grid can be unpacked anywhere, and as long as the user invoking the commands has the proper permissions on the $OMERO_PREFIX directory, it will function normally.

Running as root

One exception to this rule is that starting OMERO.grid as root may actually delegate to another user, if the “user” attribute is set on the <server/> elements in etc/grid/templates.xml (this holds only for Unix-based platforms including Apple OS X. See OMERO.grid on Windows for information on changing the server user under Windows).

See also:

OMERO sessions

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.5.2 OMERO.mail

The OMERO server has the ability to send email to any users who have a properly configured email address. OMERO system administrators can then use the omero admin email command to contact those users.

In order to activate the subsystem, minimally the omero.mail.config property will need to be activated.

6.5. Installing additional features
Minimum configuration

$ bin/omero config set omero.mail.config true

By default, this will use “localhost” as the mail server on port 25 and send as the user “omero”. Likely you will need to configure OMERO to use your actual mail server.

For example,

$ bin/omero config set omero.mail.host smtp.university.example

If authentication is required, then also configure:

$ bin/omero config set omero.mail.username USER
$ bin/omero config set omero.mail.password PASS

Setting email addresses

For the any user to receive email, a valid email address must be configured. By default, the root OMERO user will not have an email address configured. This can be done from one of the UIs or via the omero obj command:

$ bin/omero obj update Experimenter:0 email=root@university.example

Note: Using a mailing list or an alias for the root user can simplify configuration.

Enabling mail notifications

A number of “mail senders” are available for sending notifications of certain events on the server. Those available include:

- ServerUpMailSender and ServerDownMailSender which mail when the server goes up or down
- FailedLoginMailSender which can be configured to send for particular users if a bad password is used
- ObjectMailSender which can be configured to send an email under various conditions. Instances which are configured include:
  - newUserMailSender which sends an email every time a user is created
  - newCommentMailSender which sends an email every time a user’s image is commented on by another user.

To activate the senders, the etc/blitz/mail-senders.example can be copied to a file ending with ”.xml”.

OMERO.web error reporting

OMERO.web will email the users listed in the omero.web.admins whenever the application identify broken link (HTTP status code 404) or raises an unhandled exception that results in an internal server error (HTTP status code 500). This gives the administrators immediate notification of any errors. The omero.web.admins will get a description of the error, a complete Python traceback, and details about the HTTP request that caused the error.

Note: Reporting errors requires property omero.web.debug set to False and works together with OMERO.web error handling.
Other key properties

Along with omero.mail.host, a few general connection properties may be needed for your particular SMTP server:

- omero.mail.port
- omero.mail.smtp.auth
- omero.mail.smtp.starttls.enable
- omero.mail.from

Note: omero.mail.from may not be necessary but some servers may require it to match username. Regardless, it can be useful to inform users more clearly of who is getting in touch with them.

All properties can be found under the Mail section of Configuration properties glossary.

Further configuration

Finally, if the above mail configuration properties do not cover your needs, you can add your own implementation as described under Extending OMERO.server. The related property is omero.mail.bean:

```bash
$ bin/omero config set omero.mail.bean myMailImplementation
```

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.5.3 OMERO.movie

A short description on how to create movies from OMERO.

Creating a movie from OMERO

OMERO provides a script to make Mpeg or Quicktime movies from any image in the server. These movies are created by a script called makemovie.py, this script has a number of options: these include: selecting a range of Z,T planes, the channels to display. The movie can also show information overlayed over the image: z-section, scale bar and timing.

The resulting movie will then be uploaded to the server by the script and become a file attachment to the source image.

Viewing the movie

The make movie script allows you to save the movie in two different formats, a DivX encoded AVI and Quicktime movie. To view the AVI you may need to install a divX codec from DivX\(^{193}\). It should be noted that the DivX avi is normally 1/3 to 1/10 the size of the Quicktime movie.

Installing the make movie script

The make movie script currently uses the `mencoder`\(^{194}\) utility to encode the movies, this command should be in the path of the computer (icegrid node) running the script.

You can find windows installs for mencoder at http://sourceforge.net/projects/mplayer-win32/files/

We have Mac OSX installs for mencoder\(^{195}\) which were originally provided here\(^{196}\). Unzip and put the mencoder in the PATH available to the server, e.g. `/usr/local/bin/`. You may need to restart the server for this to take effect.

\(^{193}\)http://www.divx.com/
\(^{194}\)http://www.mplayerhq.hu/design7/dload.html
\(^{195}\)http://cvs.openmicroscopy.org.uk/snapshots/mencoder/mac/
\(^{196}\)http://stefpause.com/apple/mac/mplayer-os-x-10rc1-and-mencoder-binaries/
There are also macports, rpms and debs for mencoder.

Make movie also uses Pillow\(^{197}\) and numpy\(^{198}\).

### Make movie command arguments

A detailed list of the commands accepted by the script are:

- **imageId**: This id of the image to create the movie from
- **output**: The name of the output file, sans the extension
- **zStart**: The starting z-section to create the movie from
- **zEnd**: The final z-section
- **tStart**: The starting timepoint to create the movie
- **tEnd**: The final timepoint.
- **channels**: The list of channels to use in the movie (index, from 0)
- **splitView**: Should we show the split view in the movie (not available yet)
- **showTime**: Show the average time of the acquisition of the channels in the frame.
- **showPlaneInfo**: Show the time and z-section of the current frame.
- **fps**: The number of frames per second of the movie
- **scalebar**: The scalebar size in microns, if \(<=0\) will not show scale bar.
- **format**: The format of the movie to be created currently supports ‘video/mpeg’, ‘video/quicktime’
- **overlayColour**: The colour of the overlays, scalebar, time, as int(RGB)
- **fileAnnotation**: The fileAnnotation id of the uploaded movie. (return value from script)

### Platform-specific notes

#### Windows

For Windows, you can download a “MPlayer-rtm-svm-<version>” bundle from [http://sourceforge.net/projects/mplayer-win32/](http://sourceforge.net/projects/mplayer-win32/). You will need 7zip\(^{200}\) to open the bundle. Then you will need to add the directory containing “mencoder.exe” to your system path and restart.

**Note**: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 6.5.4 OMERO.scripts

OMERO.scripts are the OME version of plugins, allowing you to extend the functionality of OMERO. Official core OMERO.scripts come bundled with every OMERO.server release but you can also add new scripts you have written yourself or found via the new script sharing service\(^{201}\).

---

\(^{197}\) [http://pillow.readthedocs.org](http://pillow.readthedocs.org)  
\(^{198}\) [http://www.scipy.org/Download](http://www.scipy.org/Download)  
\(^{200}\) [http://www.7-zip.org/download.html](http://www.7-zip.org/download.html)  
\(^{201}\) [http://www.openmicroscopy.org/site/community/scripts](http://www.openmicroscopy.org/site/community/scripts)
Prerequisites

Uploading and managing scripts

OMERO.scripts user guide describes the workflow for developing and uploading scripts as an Admin. Any scripts you add to the lib/scripts/ directory as a server admin will be considered ‘trusted’ and automatically detected by OMERO, allowing them to be run on the server from the clients or command line by any of your users.

Once in the directory, scripts cannot be automatically updated and any additional ones will be lost when you upgrade your server installation. Therefore, we recommend you use a Github repository to manage your scripts. If you are not familiar with using git, you can use the GitHub app for your OS (available for Mac and Windows but not Linux). The basic workflow is:

- fork our omero-user-script repository
- clone it in your lib/scripts directory

```
  cd lib/scripts;
  git clone git@github.com:YOURGITUSERNAME/omero-user-scripts.git YOUR_SCRIPTS
```

- save the scripts you want to use into the appropriate sub-directory in your cloned location lib/scripts/YOUR_SCRIPTS

Then when you upgrade your OMERO.server installation, provided your Github repository is up to date with all your latest script versions (i.e. all your local changes are committed), you just need to repeat the `git clone` step. Those scripts will then be automatically detected by your new server installation and available for use from the clients and command line as before.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

6.5.5 OMERO.tables

OMERO.tables provide a way to efficiently store large, tabular results within OMERO. If you would like to find out more about the use of the OMERO.tables API, see OMERO.analysis

Requirements

If you would like to help test the Tables API, you will need the following installed:

- HDF5
- NumPy
- PyTables (Some packages include HDF5)

Unix

PyTables is likely available from the package repository of your Unix-flavor. This includes Mac OS X (homebrew), Debian and Ubuntu (apt-get), CentOS (yum), and SuSE (yast). Here we’ve shown manual instructions using virtualenv.

Manually

---

202http://www.openmicroscopy.org/site/support/contributing/using-git.html
203http://help.github.com/articles/set-up-git
204https://github.com/ome/omero-user-scripts
205http://www.hdfgroup.org/HDF5/release/obtain5.html
206http://numpy.sourceforge.net/numdoc/HTML/numdoc.htm
207http://pytables.github.com/downloads.html
$ virtualenv $HOME/virtualenv
$ uname -o -p
unknown GNU/Linux
$ gcc --version
gcc-4.8.real (Debian 4.8.1-9) 4.8.1
Copyright (C) 2013 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

$ wget "http://www.hdfgroup.org/ftp/HDF5/current/src/hdf5-1.8.11.tar.gz"
$ tar xzf hdf5-1.8.11.tar.gz
$ cd hdf5-1.8.11
$ ./configure --prefix=$HOME/virtualenv
$ make
$ make install
$ export LD_LIBRARY_PATH=$HOME/virtualenv/lib
$ . $HOME/virtualenv/bin/activate
$ easy_install tables

Checking that it works

After that, the following should succeed:

% python
Python 2.7.5+ (default, Aug 4 2013, 10:07:17)
[GCC 4.8.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import tables
>>> tables.test()
 ..........................................................
PyTables version: 2.4.0
HDF5 version: 1.8.11
NumPy version: 1.7.1
Numexpr version: 2.0.1 (not using Intel's VML/MKL)
ZLib version: 1.2.8 (in Python interpreter)
LZO version: 2.06 (Aug 12 2011)
BZIP2 version: 1.0.6 (6-Sept-2010)
Blosc version: 1.1.3 (2010-11-16)
Python version: 2.7.5+ (default, Aug 4 2013, 10:07:17)
[GCC 4.8.1]
Platform: linux2-x86_64
Byte-ordering: little
Detected cores: 8
...

Once the required Python libraries are installed, starting OMERO will automatically start up the OMERO.tables service; there
should be no need for further configuration or interaction.

Windows

After installing all the Windows prerequisites OMERO.tables should start up during the OMERO.server startup. It can be verified
by looking at the output of omero admin diagnostics:

(...)
Server: Tables-0 active (pid = 3176, enabled)

See also:

PostgreSQL (9.4 or higher)
6.6 Troubleshooting OMERO

6.6.1 Which user account and password do I use where?

Accounts table, including the example usernames and passwords used in the installation guides:

<table>
<thead>
<tr>
<th>Account type</th>
<th>Function</th>
<th>Username</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>Administrator/Root</td>
<td>postgres</td>
<td>root</td>
</tr>
<tr>
<td>System</td>
<td>(Database) service account</td>
<td>postgres</td>
<td>root</td>
</tr>
<tr>
<td>System</td>
<td>(OMERO) service account</td>
<td>omero_user</td>
<td>root</td>
</tr>
<tr>
<td>Database</td>
<td>Database administrator</td>
<td>db_user</td>
<td>db_password</td>
</tr>
<tr>
<td>Database</td>
<td>Database user</td>
<td>root</td>
<td>root_password</td>
</tr>
<tr>
<td>OMERO</td>
<td>OMERO administrator</td>
<td>root</td>
<td>root_password</td>
</tr>
<tr>
<td>OMERO</td>
<td>OMERO users</td>
<td>root</td>
<td>root_password</td>
</tr>
</tbody>
</table>

Note: These example usernames and passwords are provided to help you follow the installation guide examples. Do not use root_password or db_password; substitute your own passwords.

There are a total of three types of user accounts: system, database and OMERO accounts.

System accounts

These are accounts on your machine or directory server (e.g. LDAP, Active Directory). One account is used for running the OMERO server (either your own or one you created specially for running OMERO, referred to here as “omero_user”). There is also a user called the “administrator-level user” on the Windows installation page and “root-level user” on the UNIX installation page (which includes Mac OS X). A separate “postgres” user is used for running the database server. The “omero_user” account runs the OMERO server, and owns the files uploaded to OMERO. This account must have permission to write to the /OMERO/binary repository. Some operations in the install scripts require the root-level/administrator-level privileges in order to use another account to perform particular actions e.g. the “postgres” user to create a database. However the OMERO server should never be run as the root-level/administrator-level user or as the system-level “postgres” user.

Database accounts

The PostgreSQL database system contains user and administrative accounts; these are completely separate from the system accounts, above, existing only inside the database. The database administrative user (“postgres”) is the owner of all the database resources, and can create new users internal to the database. A single database account is used at run time by OMERO to talk to your database. Therefore, you must configure the database values during installation:

```sh
$ bin/omero config set omero.db.user db_user
$ bin/omero config set omero.db.pass db_password
```

Note: Do not use db_user or db_password here; substitute your own username and password.

A database user may have the same name as an account on your machine, in which case a password might not be necessary.

OMERO accounts

These accounts only exist inside the OMERO system, and are completely separate from both the system and database accounts, above. The first user which you will need to configure is the “root” OMERO user (different from any root-level Unix account). This is done by setting the password in the database script, see Database tools.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.
Other OMERO users can be created via the OMERO.web admin tool. None of the passwords have to be the same, in fact they should be different unless you are using the LDAP plugin.

### 6.6.2 Server fails to start

1. Check that you are able to successfully connect to your PostgreSQL installation as outlined on the PostgreSQL page for your OS ([Windows PostgreSQL page](https://www.postgresql.org/docs/13/windows.html) or [UNIX/Mac PostgreSQL page](https://www.postgresql.org/docs/13/unix.html)).

2. Check the permissions on your `omero.data.dir` ([/OMERO or C:/OMERO by default]) as outlined on the OMERO.server installation page for Unix/Mac users, or the OMERO.server binary repository page for Windows users.

3. Are you on a laptop? If you see an error message mentioning “node master couldn’t be reached”, you may be suffering from a network address swap. Ice does not like to have its network changed as can happen if the server is running on a laptop on wireless. If you lose connectivity to icegridnode, you may have to kill it manually via `kill PID` or `killall icegridnode` (under Unix).

4. If you see an error message mentioning “Freeze::DatabaseException” or “could not lock file: var/registry/__Freeze/lock”, your icegrid registry may have become corrupted. This is not a problem, but it will be necessary to stop OMERO and delete the `var/master` directory (e.g. `rm -rf var/master`). When restarting OMERO, the registry will be automatically re-created.

5. If you see an error message mentioning “Protocol family unavailable”, you will need to set the `Ice.IPv6` property with `omero config set Ice.IPv6 0`.

6. If you upgraded from a 5.0.2 server or older and copied the entire content of the etc/grid directory from the old server to the new server, you will need to revert the changes made to `templates.xml` to generate the new memory settings.

7. If OMERO starts up, but fails to respond to connection requests, check `netstat -a` for port 4064. If nothing is listening on 4064, you may need to specify which network interface to use. `omero config set Ice.Default.Host 127.0.0.1`, for example, would force OMERO to only listen on localhost. See [Proxy and Endpoint Syntax](https://doc.zeroc.com/display/Ice/Proxy+and+Endpoint+Syntax#ProxyandEndpointSyntax-address) for more information.

### 6.6.3 Remote clients cannot connect to OMERO installation

The Admin section of OMERO.web appears to work properly and you may or may not have created some users, but no matter what you do remote clients will not speak to OMERO. OMERO.insight gives you an error message similar to the following despite giving the correct username and password:

![Login Failure](https://trac.openmicroscopy.org/ome/ticket/7325)

This is often due to firewall misconfiguration on the machine that runs your OMERO server, affecting the ability of remote clients to locate it. Please see the [Server security and firewalls](https://www.openmicroscopy.org/ome/ticket/7325) page.

### 6.6.4 Server crashes with…

- X11 connection rejected because of wrong authentication
- X connection to localhost:10.0 broken (explicit kill or server shutdown).

---

208 https://trac.openmicroscopy.org/ome/ticket/7325
209 https://trac.openmicroscopy.org/ome/ticket/7325
210 https://trac.openmicroscopy.org/ome/ticket/12527
211 https://trac.openmicroscopy.org/ome/ticket/12527
212 https://doc.zeroc.com/display/Ice/Proxy+and+Endpoint+Syntax#ProxyandEndpointSyntax-address
OMERO uses image scaling and processing techniques that may be interfered with when used with SSH X11-forwarding. You should disable SSH X11-forwarding in your SSH session by using the `-x` flag as follows before you restart the OMERO.server:

```
ssh -x my_server.examples.com
```

### 6.6.5 OutOfMemoryError / PermGen space errors in OMERO.server logs

Out of memory or permanent generation (PermGen) errors can be caused by many things. You may be asking too much of the server. Or you may require an increase in the maximum Java heap or the permanent generation space. This can be done by modifying the configuration for your OMERO.server. See [Memory configuration](#).

Similarly, if you are finding out of memory errors in one of the other process logs (e.g. `Indexer-0.log` or `PixelData-0.log`), you might try optimizing the JVM memory settings.

Furthermore, under certain conditions access of images greater than 4GB can be problematic on 32-bit platforms due to certain bugs within the Java Virtual Machine including [Bug ID: 4724038](http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=4724038). A 64-bit platform for your OMERO.server is **HIGHLY** recommended.

### 6.6.6 Import errors

**Import error when running bin/omero**

```
Traceback (most recent call last):
File "bin/omero", line 67, in ?
    import omero.cli
ImportError: No module named omero.cli
```

If you get any import related errors while running `bin/omero`, the most likely cause is that your `PYTHONPATH` is not properly set.

- If you installed Ice globally via your package manager, make sure you included `ice-python`.
- If you installed Ice manually, e.g. under `/opt/Ice-3.5.1` you need to add `/opt/Ice-3.5.1/python` (or similar) to your `PYTHONPATH` environment variable. See the Ice installation instructions for more information.

**Too many open files**

This is most often seen as an error during importing and is caused by the number of opened files exceeding the limit imposed by your operating system. It might be due to OMERO leaking file descriptors; if you are not using the latest version, please upgrade, since a number of bugs which could cause this behavior have been fixed. It is also possible for buggy scripts which do not properly release resources to cause this error.

To view the current per-process limit, run

```
ulimit -Hn
```

which will show the hard limit for the maximum number of file descriptors (`-Sn` will show the soft limit). This limit may be increased. On Linux, see `/etc/security/limits.conf` (global PAM per-user limits configuration); it is also possible to increase the limit in the shell with

```
ulimit -n newlimit
```

providing that you are uid 0 (other users can only increase the soft limit up to the hard limit). To view the system limit, run

```
ulimit -Hn
```

---

We recommend 8K as a minimum number of files limit for production systems, with 16K being reasonable for bigger machines.

On Mac OS X, the standard ulimit will not work properly. There are several different ways of setting the ulimit, depending upon the version of OS X you are using, but the most common are to edit `sysctl.conf` or `launchd.conf` to raise the limit. However, note that both of these methods change the defaults for every process on the system, not just for a single user or service.

### Increasing the number of available filehandles via `ulimit -n`

ValueError: filedescriptor out of range in select() - this is a known issue in Python versions prior to 2.7.0. See #6201\(^{214}\) and Python Issue #3392\(^{215}\) for more details.

### Windows import errors

See Failure response on import for new databases.

#### Directory exists but is not registered

Import errors of type `Directory exists but is not registered: CheckedPath(username_id)` suggest a server-side issue under the ManagedRepository.

For production servers, this can be caused by a server crash during import or an issue at the file system level (permissions, renaming). If the ManagedRepository/username_id folder is empty, you should try removing it before trying another import.

For development servers, this may be caused by the binary directory not being cleaned after the database has been wiped.

See also:

Upload problem: Directory exists but is not registered.\(^{216}\)

import: Directory exists but is not registered: CheckedPath(\(^{217}\)

[ome-devel] Directory exists but is not registered: CheckedPath(username_id)\(^{218}\)

### 6.6.7 DropBox fails to start: failed to get session

If the main server starts but DropBox fails with the following entry in `var/log/DropBox.log`

```
2011-06-07 03:42:56,775 ERROR [ fsclient.DropBox] (MainThread) Failed to get Session:
```

then it may be that the server is taking a relatively long time to start.

A solution to this is to increase the number of retries and/or the period (seconds) between retries in `etc/grid/templates.xml`

```
<property name="omero.fs.maxRetries" value="5"/>
<property name="omero.fs.retryInterval" value="3"/>
```

---

\(^{214}\)https://trac.openmicroscopy.org/ome/ticket/6201
\(^{215}\)http://bugs.python.org/issue3392
\(^{216}\)http://www.openmicroscopy.org/community/viewtopic.php?f=5&t=7537
\(^{217}\)http://www.openmicroscopy.org/community/viewtopic.php?f=6&t=7722&p=15264&highlight=CheckedPath#p15264
\(^{218}\)http://lists.openmicroscopy.org.uk/pipermail/ome-devel/2014-October/003020.html
6.6.8 Search does not return expected results

If searching for a specific term does not return the expected results (e.g. searching for the name of a tag does not return the full list of a user’s images annotated with that tag), there are a few things that may have gone wrong. See Missing search results for more details.

6.6.9 OMERO.web issues

OMERO.web and Apache

Do not enable mod_python and mod_wsgi in the same apache process. mod_wsgi will deadlock if run in daemon mode while mod_python is enabled.

OMERO.web did not start

- If the Apache error logs contain lines of type Permission denied: access to xxx denied, you need to check the permissions of the folder and make sure it is readable and executable by the Apache user.
  
  See also:
  
  What are your LDAP requirements?²¹⁹
  
  upgrade 5.0.5 to 5.1.1 omero.web forbidden²²⁰

OMERO.web running but status says not started

If you upgraded OMERO but forgot to stop OMERO.web, processes will still be running. In order to kill stale processes by hand, run:

```
$ ps aux | grep /home/omero/OMERO.server/var/django.pid
```

Note: As Gunicorn is based on the pre-fork worker model it is enough to kill the master process, the one with the lowest PID.

OMERO.web not available HTTP 404

Consult nginx error.log for more details.

The most common problem appears when the default configuration for location / is loaded prior to omeroweb.conf

```
2016/01/01 00:00:00 [error] 1234#0: *5 "/usr/share/nginx/html/webclient/login/index.html" is not found
```

OMERO.web not responding/timeout issues

[Critical] WORKER TIMEOUT (pid:1234)

OMERO.web deployed with Gunicorn relies on the operating system to provide all of the load balancing while handling requests. Adjust the timeout using omero.web.wsgi_timeout and scale the number of omero.web.wsgi_workers starting with (2 x NUM_CORES) + 1 workers. For more details refer to Gunicorn default configuration (Unix/Linux).

²¹⁹http://www.openmicroscopy.org/community/viewtopic.php?f=5&t=14
Issues with downloading data from OMERO.web

An EXPERIMENTAL: Gunicorn advance configuration (Unix/Linux) is available for testing with nginx if you are encountering problems with downloads failing. You can also configure OMERO.web to limit downloads - refer to the OMERO.web deployment documentation and Download restrictions for further details.

OMERO.web piecharts

‘Drive space’ does not generate pie chart or ‘My account’ does not show markup picture and crop the picture options.

Error message says: ‘Piechart could not be displayed. Please check log file to solve the problem’. Please check var/log/OMEROweb.log for more details. There are a few known possibilities:

• ‘TclError: no display name and no $DISPLAY environment variable’. Turn off the compilation of TCL support in Matplotlib221.

• ‘ImportError: No module named Image’. Install Pillow222 (packages should be available for your distribution). Also double check if all of the prerequisites were installed from OMERO.web deployment (UNIX instructions or Windows instructions).

6.6.10 Troubleshooting performance issues with the clients

If you are having issues with slowdown and timeouts in the clients, there are three things to check:

• your network connection

• if you are running out of memory (processing large datasets can cause problems)

• whether your server’s HOME directory is on a network share

In the final case, two issues arise. Firstly, when performing some operations, the clients make use of temporary file storage and log directories, as described in the Client configuration section of System requirements. If your home directory is stored on a network, possibly NFS mounted (or similar), then these temporary files are being written and read over the network which can slow access down. Secondly, the OMERO.server also accesses the temporary and log folders of the user the server process is running as. As the server makes heavier use of these folders than the clients, if the OMERO.server user directory is stored on a network drive then slow performance can occur.

To resolve this, define the OMERO_TMPDIR environment variable to point at a temporary directory located on the local file system (e.g. /tmp/omero).

6.6.11 Other issues

Connection problems and TCP window scaling on Linux systems

Later versions of the 2.6 Linux kernel, specifically 2.6.17, have TCP window scaling enabled by default. If you are having initial logins never timeout or problems with connectivity in general you can try turning the feature off as follows:

# echo 0 > /proc/sys/net/ipv4/tcp_window_scaling

Server or clients print “WARNING: Prefs file removed in background...”

Nov 12, 2008 3:02:50 PM java.util.prefs.FileSystemPreferences$7 run
WARNING: Prefs file removed in background /root/.java/.userPrefs/prefs.xml
Nov 12, 2008 3:02:50 PM java.util.prefs.FileSystemPreferences$7 run
WARNING: Prefs file removed in background /usr/lib/jvm/java-1.7.0-icedtea-1.7.0.0/jre/.systemPrefs/prefs.xml

---

221 http://matplotlib.org/
222 http://pillow.readthedocs.org
These warnings (also sometimes listed as ERRORS) can be safely ignored, and are solely related to how Java is installed on your system. See http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=4751177 or this ome-users thread\(^\text{223}\) on our mailing list for more information.

Data corruption

If you are dealing with a data corruption issue, you may find the information on *PixelService resolution order for locating binary data for images* useful.

CHAPTER
SEVEN

OPTIMIZING SERVER CONFIGURATION

This chapter discusses the options for configuring OMERO.server for optimum performance and security.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

7.1 Server security and firewalls

7.1.1 General

OMERO has been built with security in mind. Various standard security practices have been adhered to during the development of the server and client including:

• Encryption of all passwords between client and server via SSL
• Full encryption of all data when requested via SSL
• User and group based access control
• Authentication via LDAP
• Limited visible TCP ports to ease firewalling
• Use of a higher level language (Java or Python) to limit buffer overflows and other security issues associated with native code
• Escaping and bind variable use in all SQL interactions performed via Hibernate

Note: The OMERO team treats the security of all components with care and attention. If you have a security issue to report, please do not hesitate to contact us using our private, secure mailing list as described on the security vulnerabilities\(^1\) page.

7.1.2 Firewall configuration

Securing your OMERO system with so called firewalling or packet filtering can be done quite easily. By default, OMERO clients only need to connect to two TCP ports for communication with your OMERO.server: 4063 (unsecured) and 4064 (SSL). These are the IANA\(^2\) assigned ports for the Glacier2 router from ZeroC\(^3\). Both of these values, however, are completely up to you, see SSL below.

Important OMERO ports:

• TCP/4063
• TCP/4064

If you are using OMERO.web, then you will also need to make your HTTP and HTTPS ports available. These are usually 80 and 443.

Important OMERO.web ports:

1^1\)\) http://www.openmicroscopy.org/info/vulnerabilities/
2^2\) http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml
3^3\) https://zeroc.com
• TCP/80
• TCP/443

Example OpenBSD firewall rules

block in log on $ext_if from any to <omero_server_ip>
pass in on $ext_if proto tcp from any to <omero_server_ip> port 4063
pass in on $ext_if proto tcp from any to <omero_server_ip> port 4064
pass in on $ext_if proto tcp from any to <omero_server_ip> port 443
pass in on $ext_if proto tcp from any to <omero_server_ip> port 80

Example Linux firewall rules

iptables -P INPUT drop
iptables -A INPUT -p tcp --dport 4063 -j ACCEPT
iptables -A INPUT -p tcp --dport 4064 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
...

7.1.3 Passwords

The passwords stored in the password table are salted and hashed, so it is impossible to recover a lost one, instead a new one must be set by an admin.

If the password for the root user is lost, the only way to reset it (in the absence of other admin accounts) is to manually update the password table. The bin/omero command can generate the required SQL statement for you:

$ bin/omero db password
Please enter password for OMERO root user:
Please re-enter password for OMERO root user:
UPDATE password SET hash = 'PJueOtwuTPHB8Nq/1rFVxg==' WHERE experimenter_id = 0;

Current hashed password:

$ psql mydatabase -c "select * from password"
  experimenter_id | hash
-----------------+--------------------------
  0 | Xr4ilOzQ4PCOq3aQ0qbuaQ== (1 row)

Change the password using the generated SQL statement:

$ psql mydatabase -c "UPDATE password SET hash = 'PJueOtwuTPHB8Nq/1rFVxg==' WHERE experimenter_id = 0;"
UPDATE 1

7.1.4 Java key- and truststores.

If your server is connecting to another server over SSL, you may need to configure a truststore and/or a keystore for the Java process. This happens, for example, when your LDAP server uses SSL. See the LDAP plugin for information on how to configure the LDAP URLs. As with all configuration properties, you will need to restart your server after changing them.
To do this, you will need to configure several server properties, similar to the properties you configured during installation (Windows).

- **truststore path**

  ```
  bin/omero config set omero.security.trustStore /home/user/.keystore
  ```

  A truststore is a database of trusted entities and their associated X.509 certificate chains authenticating the corresponding public keys. The truststore contains the Certificate Authority (CA) certificates and the certificate(s) of the other party to which this entity intends to send encrypted (confidential) data. This file must contain the public key certificates of the CA and the client’s public key certificate.

  If you don’t have one you can create it using the following:

  ```
  openssl s_client -connect {{host}}:{{port}} -prexit < /dev/null | openssl x509 -outform PEM | keytool -import -alias ldap -storepass {{password}} -keystore {{truststore}} -noprompt
  ```

- **truststore password**

  ```
  bin/omero config set omero.security.trustStorePassword secret
  ```

- **keystore path**

  ```
  bin/omero config set omero.security.keyStore /home/user/.mystore
  ```

  A keystore is a database of private keys and their associated X.509 certificate chains authenticating the corresponding public keys.

  A keystore is mostly needed if you are doing client-side certificates for authentication against your LDAP server.

- **keystore password**

  ```
  bin/omero config set omero.security.keyStorePassword secret
  ```

### 7.1.5 SSL

Especially if you are going to use LDAP authentication to your server, it is important to encrypt the transport channel between clients and the Glacier2 router to keep your passwords safe.

By default, all logins to OMERO occur over SSL using an anonymous handshake. After the initial connection, communication is un-encrypted to speed up image loading. Clients can still request to have all communications encrypted by clicking on the lock symbol. An unlocked symbol means that non-password related activities (i.e. anything other than login and changing your password) will be unencrypted, and the only critical connection which is passed in the clear is your session id.

Administrators can configure OMERO such that unencrypted connections are not allowed, and the user’s choice will be silently ignored. The SSL and non-SSL ports are configured in the `etc/grid/default.xml` and `etc/grid/windefault.xml` files, and as described above, default to 4064 and 4063 respectively, and can be modified using the `Ports` configuration properties. For instance, to prefix all ports with 1, use `omero.ports.prefix`:

```
$ bin/omero config set omero.ports.prefix 1
```
7.2 LDAP authentication

LDAP\(^4\) is an open standard for querying and modifying directory services that is commonly used for authentication, authorization and accounting (AAA). OMERO.server supports the use of an LDAP server to query (but not modify) AAA information for the purposes of automatic user creation.

This allows OMERO users to be automatically created and placed in groups according to your existing institution policies. This can significantly simplify your user administration burden. Note that OMERO has its own concept of “groups” that is quite distinct from LDAP groups.

The OMERO.server LDAP implementation can handle a number of use cases. For example:

- Allow every “inetOrgPerson” under `omero.ldap.base` to login
- but restrict access based upon an arbitrary LDAP filter, e.g.

\[
\text{omero.ldap.user_filter=(memberOf=cn=someGroup,ou=Lab,o=College)}
\]

- and add that user to some number of groups, e.g.

\[
\text{omero.ldap.new_user_group=query:(member=@{dn})}
\]

7.2.1 How it works

On login, the username provided is searched for in OMERO. If the name does not exist, then the LDAP plugin is queried for a username matching the system-wide user filter. If such an LDAP entry exists and the password matches, a new user with the given username is created, and the user is added to any groups which match the `new_user_group` setting.

On subsequent logins, the user filter and the password are again checked against the LDAP server, and if there is no longer a match, login is refused. If you would prefer to only have the `user_filter` applied during user creation and not on every login, see Legacy password providers.

You can take existing non-LDAP users and ‘upgrade’ them to using LDAP with the OMERO command line tool, see Converting non-LDAP users to LDAP authentication. You can also use `omero ldap create` to add an ldap user to OMERO groups without requiring them to log in first, see User/group management for details.

7.2.2 LDAP properties

The LDAP plugin is configured via several configuration properties, all starting with `omero.ldap` (see LDAP).

Some of the property values are passed directly to the underlying LDAP library (Spring LDAP\(^5\)), which in turn makes use of the Java API. OMERO does not modify the error messages thrown by the library or by Java, so please consult the appropriate documentation to diagnose any low-level problems.

Note: Please remember that once a change has been made, a server restart will be needed.

---


\(^5\)http://projects.spring.io/spring-ldap/
Minimum configuration

The following properties are the minimum requirements for logging in to OMERO using LDAP.

omero.ldap.config=true
omero.ldap.url=ldap://localhost:389
omero.ldap.username=
omero.ldap.password=
omero.ldap.base=ou=example,o=com

After having configured your connection, you can turn LDAP on and off between restarts by setting `omero.ldap.config` to false. The `base` property determines where in the LDAP tree searches will begin. No users or groups will be found if they are not under the base provided.

User lookup

Two user properties are used to look up users by login name and, if necessary, create new users based on the information in LDAP.

`omero.ldap.user_filter=(objectClass=person)`
`omero.ldap.user_mapping=omeName=cn,firstName=givenName,lastName=sn,email=mail,institution=department,middleName=middleName`

`omero.ldap.user_filter` will be AND’ed to the username query, and can contain any valid LDAP filter string. The username query is taken from the LDAP attribute which gets mapped to “omeName” by `omero.ldap.user_mapping`. Here, the “cn” is mapped to “omeName”, so the username query is `(cn=[login name])`. The final query is `(&(objectClass=person)(cn=[login name]))`, which must return a single result to be considered valid.

Group lookup

Three group properties are all concerned with what groups a user will be placed in on creation.

`omero.ldap.group_filter=(objectClass=groupOfNames)`
`omero.ldap.group_mapping=name=cn`
`omero.ldap.new_user_group=default`

The group filter and group mapping work just as the user filter and mapping do, in that the group name query will be AND’d with the `group_filter`. In this case, the final query would be `(&(objectClass=groupOfNames)(cn=[group name]))`. However, these properties may not be used depending on the value of `new_user_group`, which can have several different values:

- If not prefixed at all, then the value is simply the name of a group which all users from LDAP should be added to.
- If prefixed with :ou:, then a user’s last organizational unit (OU) will be used as his or her group. For example, the user with the DN “cn=frank,ou=TheLab,ou=LifeSciences,o=TheCollege” will be placed in the group “TheLab”.
- If prefixed with :attribute:, then the rest of the string is taken to be an attribute all of whose values will be taken as group names. For example, `omero.ldap.new_user_group=:attribute:memberOf` would add a user to all the groups named by memberOf. You can prefix this value with filtered_ to have the `group_filter` applied to the attribute values, i.e. `:filtered_attribute:memberOf` will mean that only the values of memberOf which match `group_filter` will be considered. An example value of the `memberOf` attribute would be: `CN=mygroup,OU=My Group,OU=LabUsers, DC=openmicroscopy,DC=org`
- If prefixed with :dn_attribute:, then the rest of the string is taken to be an attribute all of whose values will be taken as group distinguished names. For example, `omero.ldap.new_user_group=:dn_attribute:memberOf` would add a user to all the groups named by memberOf, where the name of the group is mapped via `group_mapping`. You can prefix this value with filtered_ to have the `group_filter` applied to the attribute values, i.e. `:filtered_dn_attribute:memberOf` will mean that only the values of memberOf which match `group_filter` will be considered. An example value of the `memberOf` attribute would be: `CN=mygroup,OU=My Group,OU=LabUsers, DC=openmicroscopy,DC=org`
Note that if an attribute specified in `omero.ldap.group_mapping` does not constitute a part of the Distinguished Name (DN) as determined by your LDAP server then it can only be found by using `:attribute:` or `:filtered_attribute:` instead. Typical attributes that comprise the DN are: DC, CN, OU, O, STREET, L, ST, C and UID.

- If prefixed with `:query:`, then the rest of the value is taken as a query to be AND’ed to the group filter. In the query, values from the user such as `"@{cn}"`, `"@{email}"`, or `"@{dn}"` can be used as place holders.
- If prefixed with `:bean:`, then the rest of the string is the name of a Spring bean which implements the NewUserGroupBean interface. See the developer documentation `LDAP plugin design` for more info.

### Compound Filters

**Note:** OMERO uses standard RFC 2254 LDAP filters, so they must conform to that syntax and are only able to do what those filters can do. You can test the filters via ldapssearch on your OMERO server (assuming you have the OpenLDAP binaries installed).

If you are using OpenLDAP make sure your directory has the `memberOf` attribute correctly configured. Some versions of ApacheDS do not support `memberOf` at all.

Both the `user_filter` and the `group_filter` can contain any valid LDAP filter string. These must be a valid filter in themselves. e.g.

```java
omero.ldap.user_filter=|(ou=Queensland Brain Institute)(ou=Ageing Dementia Research))
```

The “|” operator (read: “OR”) above allows members of two organizational units to login to OMERO. Expanding the list allows concentric “rings” of more and more OU’s granular access to OMERO.

```java
omero.ldap.group_filter=(&{objectClass=groupOfNames} (mail=omero.flag))
```

The “&” operator (read: “AND”) produces a filter that will only match groups that have the `mail` attribute set to the value `omero.flag`. When combined with the `group_mapping`, the final query would be `(&(&{objectClass=groupOfNames} (mail=omero.flag)) (cn=[group name]))`

This is the same as the query `(&{objectClass=groupOfNames} (mail=omero.flag) (cn=[group name]))` but setting `group_filter` to `{objectClass=groupOfNames} (mail=omero.flag)` is not valid as that is not a valid filter on its own.

To restrict the list of groups to just the ones returned by the above query, the following setting is also required to remove unmatched groups:

```java
omero.ldap.new_user_group=:filtered_dn_attribute:memberOf
```

### 7.2.3 Case sensitivity

By default, the LDAP plugin is case-sensitive i.e. it will treat the usernames JSmith and jsmith as two different users. You can remove case sensitivity using:

```bash
bin/omero config set omero.security.ignore_case true
```

**Warning:** Enabling this option will affect all usernames in your OMERO system. It is the system administrator’s responsibility to handle any username clashes which may result. Making all usernames lowercase is recommended.

---

6http://www.faqs.org/rfcs/rfc2254.html
7.2.4 LDAP over SSL

If you are connecting to your server over SSL, that is, if your URL is of the form `ldaps://ldap.example.com:636` you may need to configure a key and trust store for Java. See the Server security and firewalls page for more information.

7.2.5 Synchronizing LDAP on user login

This feature allows for LDAP to be considered the authority on user/group membership. With the following setting enabled, each time a user logs in to OMERO their LDAP groups will be read from the LDAP server and reflected in OMERO:

```
bin/omero config set omero.ldap.sync_on_login true
```

Admin actions carried out in the clients may not survive this synchronization e.g. if an admin has removed an LDAP user from an LDAP group in the UI, the user will be re-added to the group when logging in again after the synchronization.

**Note:** This applies to groups created by LDAP in OMERO 5.1.x. Groups created in older versions of OMERO will not be registered as LDAP groups if you have manually altered their membership, even if the membership now matches the LDAP group.

```
bin/omero ldap setdn true --group-name $NAME
```
can be used to make these previous OMERO groups into LDAP groups.

7.2.6 Legacy password providers

The primary component of the LDAP plugin is the LdapPasswordProvider, which is responsible for creating users, checking their passwords, and adding them to or removing them from groups. The default password provider is the chainedPasswordProvider which first checks LDAP if LDAP is enabled, and then checks JDBC. This can explicitly be enabled by executing the system admin command:

```
bin/omero config set omero.security.password_provider chainedPasswordProvider
```

When the LDAP password provider implementation changes, previous versions can be configured as necessary.

- **chainedPasswordProviderNoSalt**
  The chainedPasswordProviderNoSalt uses the version of the JDBC password provider without password salting support as available in the OMERO 4.4.x series. To enable it, use:

  ```
  bin/omero config set omero.security.password_provider chainedPasswordProviderNoSalt
  ```

- **chainedPasswordProvider431**
  With the 431 password provider, the user filter is only checked on first login and not kept on subsequent logins. This allows for an OMERO admin to change the username of a user in omero to be different than the one kept in LDAP. To enable it, use:

  ```
  bin/omero config set omero.security.password_provider chainedPasswordProvider431
  ```

See also:

OMERO.server installation Installation guide for OMERO.server under UNIX-based platforms

Server security and firewalls Security pages for OMERO.server

LDAP plugin design Developer documentation on extending the LDAP plugin yourself.

What are your LDAP requirements? Forum discussion if you have LDAP requirements that are not covered by the above configuration
JNDI referrals documentation  http://docs.oracle.com/javase/jndi/tutorial/ldap/referral/jndi.html

7.2.7 Active Directory

Active Directory\(^8\) (AD) supports a form of LDAP and can be used by OMERO like most other directory services.

In AD, the Domain Services\(^9\) (DS) ‘forest’ is a complete instance of an Active Directory which contains one or more domains. Querying a particular Domain Service will yield results which are local to that domain only. In an environment with just one domain it is possible to use the default configuration instructions for OMERO LDAP. If there are multiple domains in the forest then it is necessary to query the Global Catalogue to enable querying across all of them.

Global Catalogue

In an AD DS forest, a Global Catalogue\(^10\) provides a central repository of all the domain information from all of the domains. This can be queried in the same way as a specific Domain Service using LDAP, but it runs on different ports; 3268 and 3269 (SSL).

- LDAP AD Global Catalogue server URL string

```
bin/omero config set omero.ldap.urls ldap://ldap.example.com:3268
```

Note: A SSL URL above should look like this: ldaps://ldap.example.com:3269

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

7.3 Performance and monitoring

Once you have your OMERO server running and secured, a second critical step will be tuning various configuration parameters in order to get optimal performance.

7.3.1 Memory configuration

OMERO should automatically configure itself to take advantage of the physical memory installed on a system whilst leaving room for other services. You may wish to override the defaults on a production server, for instance if your machine is solely dedicated to running OMERO you can increase the amount of memory that OMERO will use. You may also need to modify your settings if you are seeing out-of-memory errors when dealing with certain types of images.

A number of configuration properties starting with `omero.jvmcfg` control the calculation of how much memory to allocate to various OMERO services on startup, most importantly:

- blitz
- indexer
- pixeldata

Configuration properties

Configuration properties can either be applied to all three service types at the same time by omitting the service type (e.g. `omero.jvmcfg.strategy`) or to each individually by including it (e.g. `omero.jvmcfg.strategy.blitz`).

For example, the default, PercentStrategy, is equivalent to making the call:

\(^8\)http://en.wikipedia.org/wiki/Active_Directory  
$ bin/omero config set omero.jvmcfg.strategy percent

This could be changed to use the ManualStrategy for all servers:

$ bin/omero config set omero.jvmcfg.strategy manual

**Strategies**

A couple of strategies are available for calculating the effective JVM settings from the provided configuration properties.

**PercentStrategy**  Default. Reads the `percent` configuration property which can also be set globally or on a service-type basis. This percentage (0-100) of the system memory is used for the process, subject to minimum and maximum limits which can be changed. `omero.jvmcfg.system_memory`, `omero.jvmcfg.min_system_memory`, and `omero.jvmcfg.max_system_memory` are all used for defining the system memory seen. The default percentages are: blitz and pixeldata 15%, indexer 10%. If `omero.jvmcfg.perm_gen` or `omero.jvmcfg.heap_size` are explicitly set, they will be used directly as with the ManualStrategy.

**ManualStrategy**  Simply provides the values given as the JVM settings. If no value is set for a particular configuration property, then the default is used: `heap_size=512m` and `perm_gen=128m` These values are equivalent to the defaults in OMERO 5.0.2 and earlier.

**Examples**

$ bin/omero config set omero.jvmcfg.percent.blitz 50

would raise the blitz heap size to 50% of the system memory seen.

$ bin/omero config set omero.jvmcfg.system_memory 24000

would set the system memory seen to 24GB regardless of the actual amount of memory present in the system. The PercentageStrategy would use this as the basis for setting the Java heap sizes for all services.

$ bin/omero config set omero.jvmcfg.max_system_memory 64000

would raise the maximum system memory seen by an OMERO installation to 64000MB of system memory. Assuming there was at least 64000MB of memory installed blitz would default to using 9600MB.

$ bin/omero config set omero.jvmcfg.strategy.indexer manual
$ bin/omero config set omero.jvmcfg.heap_size.indexer 2000

would set the indexer heap size to 2000MB without modifying the settings for the other services.

**Tips**

View the memory settings that will apply to a newly started server.

$ bin/omero admin jvmcfg

After modifying any memory settings, be sure to restart your server.
$ bin/omero admin restart

See also:

Grid configuration Section of the advanced server configuration documentation describing etc/grid/templates.xml.

### 7.3.2 Monitoring

In addition to watching the OMERO log files, the JVM itself provides a number of tools that you can use to determine the health of your server. JVisualVM\(^{11}\), for example, can be used to visualize the memory use of each JVM:

![JVM Memory Visualization](image)

You will need to have the PID (process ID) for the service you want to monitor, which will usually be the main Blitz process. You can find the PID either via `omero admin diagnostics` or alternatively via the `jps` command found in the JDK.

Another tool, JConsole\(^{12}\), also provides access to the memory statistics for your JVM, but also lists the JMX (Java Management Extensions) management beans which provide extensive information about the running process. Information includes the number of queries that have been run, the number of open file handles, the system properties that were set on startup, and much more. Further, the `ome.system.metrics` package makes use of JMX to expose further properties.

With further configuration, JMX properties can also be accessed remotely which can be very useful for monitoring your server with Check_MK\(^ {13}\), Nagios\(^ {14}\), Zenoss\(^ {15}\), or similar. However, care must be taken to protect the exposed ports.

**Note:** The commands above require the Java JDK (Java Development Kit) as opposed to the JRE (Java Runtime Environment).

\(^{11}\)http://visualvm.java.net/
\(^{12}\)http://openjdk.java.net/tools/svc/jconsole/
\(^{13}\)http://mathias-kettner.de/check_mk.html
\(^{14}\)http://www.nagios.org/
\(^{15}\)http://www.zenoss.com/
7.3.3 Metrics

Building on top of Coda Hale’s Metrics\(^\text{16}\) library, OMERO provides the ome.system.metrics package which measures a number of internal events and makes them available both via JMX as described under Monitoring but also prints them to the log files.

By default, these values are printed to each of the JVM-based log files (e.g. `var/log/Blitz-0.log`, `var/log/Indexer-0.log`, etc) once per hour. This value can be configured via `omero.metrics.slf4j_minutes`. A typical value might look like:

```
11:28:18,923 INFO [ ome.system.metrics] (r-thread-1) type=TIMER, name=ome.services.fulltext.FullTextIndexer.batch...
```

Values include basic statistics (count, min, max, mean, etc.) as well as 75th, 90th, 95th, etc percentiles. Further, the rate over the last minute, the last 5 minutes, and the last 15 minutes is provided (m1, m5, m15). For example:

- `count=3601`
- `min=0.41...
- `max=7.85...
- `mean=0.94...
- `stddev=0.31...
- `median=0.96...

\(^{16}\text{http://metrics.dropwizard.io}\)
Useful metrics include:

- `ch.qos.logback.core.Appender.error` The number and rate of errors that have been logged. (All services)
- `jvm.fileDescriptorCountRatio` The ratio of used to available file descriptors. (All services)
- `ome.services.eventlogs.EventLogQueue.priorityCount` The number of items in the queue. (Indexer-only)
- `ome.io.nio.PixelsService.minmaxTimes` Time taken to generate min/max values per plane. (PixelData-only)
- `ome.io.nio.PixelsService.tileTimes` Time taken to generate tiled-pyramids for a big image. (PixelData-only)

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 7.4 Search and indexing configuration

#### 7.4.1 How Indexing works

Indexing is not driven by the user, but happens automatically in the background and can be controlled by a number of settings listed under Search. The indexer runs periodically as defined by `omero.search.cron` and parses the latest batch of new or modified objects in the database.

Upon successful completion, the persistent count in the configuration table will be incremented.

```sql
omero=# select value from configuration where name = 'PersistentEventLogLoader.v2.current_id';
value
-------
30983
(1 row)
```

Note: Presence of more than one `PersistentEventLogLoader.*` value in your database indicates that you have run indexing with multiple versions of the server. This is fine. To allow a new server version to force an update, the configuration key may be changed. For example, `PersistentEventLogLoader.current_id` became `PersistentEventLogLoader.v2.current_id` in `a5cb64a`17.

#### 7.4.2 Missing search results

If you are having any difficulty with search results not appearing timely, first you should start by checking the health of the Indexer-0 process:

[17]https://github.com/openmicroscopy/openmicroscopy/commit/a5cb64a
• Check the server’s log directory for a file named `Indexer-0.log` and monitor its progress (e.g. using `tail` or similar). If messages of the format:

```
INFO [ ome.services.fulltext.FullTextIndexer ] (3-thread-2) INDEXED 2 objects in 1 batch(es) [2483 ms.]
```

are periodically being appended to the log file, then your indexer process may be running behind. You can either wait for it to catch up, or try increasing the search batch size in order to speed processing. See the section on the `omero.search.batch` setting for more information.

• If there are no updates to the `Indexer-0.log` file even when new images, tags, files, etc. are added to the server, then it is possible that the Indexer process has become stuck. It is possible to force a restart of the indexer using the `IceGrid Tools` like so:

```
> bin/omero admin ice
> omero admin ice
Ice 3.5.1 Copyright (c) 2003-2013 ZeroC, Inc.
>>> server list
Blitz-0
DropBox
FileServer
Indexer-0
...
>>> server stop Indexer-0
```

You do not need to manually re-start the Indexer, as IceGrid will handle the creation of a new Indexer process automatically. In case neither of the above seems to be the case, then your indexer is running normally and more likely your index has been corrupted. You will need to **re-index** OMERO. Reasons why this might have occurred include:

• Missing search terms are part of a very large text file. If the indexer’s maximum file size limit is reached, a file may not be indexed. See the section on the `omero.search.max_file_size` setting for more information on increasing this limit.

• A bug in Lucene prior to OMERO 5.0.1 caused some documents to be “sealed” in that old search terms would return the document, but newer terms would **not**.

### 7.4.3 Re-indexing

#### Background re-indexing

Under most circumstances, you should be able to re-index the database while the server is still running. If you need to make any adjustments to the server configuration or the process heap size, first shut the server down and make these changes before restarting the server. Use the following steps to initiate a re-indexing.

• Disable the search indexer process and stop any currently running indexer processes:

```
$ bin/omero admin reindex --prepare
```

• Remove the existing search Indexes by deleting the contents of the `FullText` subdirectory of your `omero.data.dir`:

```
$ bin/omero admin reindex --wipe
```

• Reset the indexer’s progress counter in the database:

```
$ bin/omero admin reindex --reset 0
```

• Re-enable/restart the indexer process:
$ bin/omero admin reindex --finish

Depending on the size of your database, it may take the indexer some time to finish re-indexing. During this time, your OMERO server will remain available for use, however the search functionality will be degraded until the re-indexing is finished. See *Monitoring re-indexing* for information on how long this should take.

**Note:** Once you wipe your full-text directory, searches will return fewer or no results until re-indexing is complete.

### Off-line re-indexing

It is also possible to re-index the database with the server off-line. First, shutdown the OMERO server as normal and make any adjustments to the configuration that need to be made. Clear the contents of the FullText directory and reset the indexing’s progress counter as above:

```
$ bin/omero admin reindex --wipe
$ bin/omero admin reindex --reset 0
```

Then run the off-line re-indexing command:

```
$ bin/omero admin reindex --foreground
```

Re-indexing the database in off-line mode will use a 1 GB heap by default, but this can be specified on the command-line with the `--mem` argument:

```
$ bin/omero admin reindex --foreground --mem=2g
```

Other search configuration properties from *Search* can be set for the processing by setting the `JAVA_OPTS` environment variable:

```
$ JAVA_OPTS="-Domero.search.max_partition_size=100000" bin/omero admin reindex --foreground
```

Once foreground indexing is complete, re-enable the background indexer as above:

```
$ bin/omero admin reindex --finish
```

### Monitoring re-indexing

During re-indexing, it is possible to estimate the percent indexed using the following SQL command:

```
OMERO=> select 'At ' || current_timestamp(0) || ' ', Percent indexed: ' || trunc(((select count(*) from eventlog el, ... as int) and (c.name like 'PersistentEventLogLoader%')) * 1.0) / (select count(*) from eventlog) * 100, 2) || '%';
```

```sql
At 2014-06-14 07:54:37+00, Percent indexed: 70.90%
```

This value is also logged periodically both when re-indexing in the background and the foreground and is available via JMX. See *Metrics* for more information.

The overall re-indexing performance depends significantly on the memory settings and the size of the repository to index. The following table provides estimates of the process duration based on re-indexing of existing production servers of various sizes:

---

7.4. Search and indexing configuration 228
**7.5 FS configuration options**

### 7.5.1 Background

Users import their image files to the OMERO.fs server. The contents of these files are kept intact by the server and the import process preserves the files’ path and name (at least within the rules of `omero.fs.repo.path_rules` below), so that OMERO.fs can become a trusted repository for the master copy of users’ data. While the default server configuration from `Configuration properties glossary` should typically suffice, `omerocfg config set` may be used to adjust settings related to file uploads. These settings are explained below.

### 7.5.2 Repository location

Several properties determine where FS-imported files are stored:

- `omero.data.dir` - singleton property (i.e. once globally) which points to the legacy repository location for OMERO. For OMERO to run on multiple systems, the contents of this directory must be on a shared volume.

- `omero.managed.dir` - singleton property which points to the default ManagedRepository. In an OMERO install in which there is only one Blitz server, this will be the only repository. This need not be located under `omero.data.dir` but is by default.

- `omero.repo.dir` (experimental) - value passed to all non-legacy, standalone repositories. This is not actively used, but would allow hosting repositories on multiple physical systems without the need for a shared volume. For example, after running `bin/omero admin start` on the main machine, it would be possible to launch nodes on various machines via `bin/omero node start fs-B`, `bin/omero node start fs-C`, etc. Each of these would pass a different `omero.repo.dir` value to its process.

### 7.5.3 Template path

When files are uploaded to the managed repository, a parent directory is created to receive the upload. A multi-file image has all its files stored in the same parent directory, though they may be in different subdirectories of that parent to mirror the original directory structure before upload. The `omero.fs.repo.path` setting defines the creation of that parent directory. It is this value which makes the ManagedRepository “managed”.

#### Path naming constraints

There is some flexibility in how this parent directory is named. The constraints are:

- The path components (individual directories in the path) must be separated by `/` characters, **even on Windows systems**.

- A path component separator may be written as `//` only if followed by at least one more path component. In this case:
  - The server ensures that the path components preceding the `//` are owned by the `root` user.
  - Any newly created path components following the `//` are owned by the user who owns the images.

---

• If // is present then all newly created path components are owned by the user who owns the images.

• The path must be unique for each import. It is for this reason that the %time% term expands to a time with millisecond resolution.

• To avoid confusion with the expansion terms enumerated below, avoid other uses of the % character in path components.

In the above, ownership of path components is in the context of OMERO users accessing the OMERO managed repository through its API. It does not relate to operating system users’ permissions for the underlying filesystem.

Expansion terms

Special terms may be used within path components: these are replaced with text that depends on the import.

For any directory in the template path

%userId% expands to the user’s numerical ID
%user% expands to the user’s name
%institution% expands to the user’s institution name; this path component is wholly omitted if the user has no institution set
%institution:default% expands to the user’s institution name, or to the supplied “default” if the user has no institution set; for instance, %institution:State College of Florida, Manatee-Sarasota% is permitted
%groupId% expands to the OMERO group’s numerical ID
%group% expands to the OMERO group’s name
%perms% expands to the group’s six-character permissions string, for example rw---- for a private group
%year% expands to the current year number, for example 2014
%month% expands to the current month number, zero-padded, for example 08
%monthname% expands to the current month name, for example August
%day% expands to the current day number in the month, zero-padded, for example 04
%sessionId% expands to the session’s numerical ID
%session% expands to the session key (UUID) of the session, for example 6c2dae43-cfad-48ce-af6f-025569f9e6df

For user-owned directories only

These expansion terms may not precede // in the template path.
%time% expands to the current time, in hours, minutes, seconds, milliseconds, for example 13-49-07.727
%hash% expands to an eight-digit hexadecimal hash code that is constant for the set of files being imported, for example 0554E3A1
%hash: digits% expands as %hash%, where digits is a comma-separated list of how many digits of the hash to use in different subdirectories; for example, hash-%hash:3,3,2% expands to a form like hash-123/456/78
%increment% expands to an integer that increases consecutively so as to create the next new directory, for example using inc-%increment% with preexisting directories up to inc-24 would expand to inc-25
%increment: digits% expands as %increment% where digits specifies a minimum length to which to zero-pad the integer, for example using inc-%increment:3% with preexisting directories up to inc-024 would expand to inc-025
%subdirs% expands to nothing until the preceding directory has more than one thousand entries, in which case it expands to an integer that increases consecutively to similarly limit the entry count in subdirectories; applies recursively to extend the number of path components as needed, so, using example/below-%subdirs% in the path, with example/below-000 to example/below-999 all “full”, three-digit subdirectories below those are created, such as example/below-123/456
%subdirs:digits% expands as %subdirs% where digits specifies to how many digits %subdirs% may expand for each path component: for example, example/%subdirs:4%-below allows ten thousand directory entries in example before creating example/1234-below and, much later, example/1234-below/5678

No more than one of either %subdirs% or %increment% may be used in any one path component, although they may each be used many times in the whole path.

### 7.5.4 Legal file names

Although OMERO.fs attempts to preserve file naming, the server’s operating system or file system is likely to somehow constrain what file names may be stored by OMERO.fs. This is of particular concern when a user may upload from a more permissive system to a server on a less permissive system, or when it is anticipated that the server itself may be migrated to a less permissive system. The server never accepts Unicode control characters in file names.

The omero.fs.repo.path_rules setting defines the combination of restrictions that the server must apply in accepting file uploads. The restrictions are grouped into named sets:

- **Windows required** prohibits names with the characters ",", ",*, ",/,,",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",","
7.6.1 Modifying the application descriptors

When you run `omero admin start` without any other arguments, it looks up the default application descriptor for your platform:

```
$ bin/omero admin start
No descriptor given. Using etc/grid/default.xml
Waiting on startup. Use CTRL-C to exit
```

The “start” and “deploy” command, however, take several other parameters:

```
$ bin/omero admin start --help
usage: bin/omero admin start [-h] [-u USER] [file] [targets [targets ...]]
```

Start icegridnode daemon and waits for required components to come up, i.e. status == 0

If the first argument can be found as a file, it will be deployed as the application descriptor rather than etc/grid/default.xml. All other arguments will be used as targets to enable optional sections of the descriptor.

Positional Arguments:
- `file` Application descriptor. If not provided, a default will be used
- `targets` Targets within the application descriptor which should be activated.

If a file is passed in as the first argument, then that application descriptor as opposed to etc/grid/default.xml will be used. You can also modify the default application descriptors in place.

Note: The largest issue with using your own application descriptors or modifying the existing ones is that they tend to change between versions, and there is no facility for automatically merging your local changes. You should be prepared to re-make whatever changes you perform directly on the new files.

7.6.2 Targets

Targets are elements within the application descriptors which can optionally turn on configuration. The target is only applicable until the next invocation of `omero admin start` or `omero admin deploy`.

Note: You must remember to always apply the targets on each `omero admin` command. If not, the target will not be removed. Therefore, they are often better used for debugging purposes; however, as opposed to alternative application descriptors, using the pre-existing targets should not require any special effort during upgrades.

Debugging

```
<properties id="PythonServer">
  <property name="Ice.ImplicitContext" value="Shared"/>
  <!-- Default logging settings for Python servers. -->
  <property name="omero.logging.timedlog" value="False"/>
  <property name="omero.logging.logsize" value="5000000"/>
  <property name="omero.logging.lognum" value="9"/>
  <property name="omero.logging.level" value="20"/>
  <target name="debug">
    <property name="omero.logging.level" value="10"/>
  </target>
</properties>
```

Here, the “debug” target allows increasing the logging output of the Python servers without modifying any files.
JMX configuration

```xml
<server-template id="BlitzTemplate">
  <parameter name="index"/>
  <parameter name="config" default="default"/>
  <parameter name="jmxhost" default=""/>
  <parameter name="jmxport" default="3001"/>
  ...
  <target name="jmx">
    <!-- Be sure to understand the consequences of enabling JMX.
    It allows calling remote methods on your JVM -->
    <option>-Dcom.sun.management.jmxremote=${jmxhost}</option>
    <option>-Dcom.sun.management.jmxremote.port=${jmxport}</option>
    <option>-Dcom.sun.management.jmxremote.authenticate=false</option>
    <option>-Dcom.sun.management.jmxremote.ssl=false</option>
  </target>
</server-template>
```

The JMX target activates the monitoring of the Blitz server via JMX. If you need to modify the “jmxport” or “jmxhost” variables, you will need to do so directly in the application descriptor XML.

### 7.6.3 Changing ports / multiple servers on a single host

By modifying the default OMERO ports, it is possible to run multiple OMERO servers on the same physical machine. All port numbers can be adjusted using the relevant configuration properties.

To run multiple servers on a single host, the easiest approach is to prefix all ports (SSL, TCP, registry) using `omero.ports.prefix`:

```
# First server
cd ~/OMERO.server-1
bin/omero admin start

# Second server
cd ~/OMERO.server-2
bin/omero config set omero.ports.prefix 1
bin/omero admin start

# Third server
cd ~/OMERO.server-3
bin/omero config set omero.ports.prefix 2
bin/omero admin start
```

Clients will need to use the appropriate port (4064, 14064 or 24064) to connect to OMERO.

**See also:**

*SSL*  Section of the *Server security and firewalls* page.

### 7.6.4 Extending OMERO

Finally, if configuration does not suffice, there are also options for extending OMERO with your own code. These are described on the development site under *Extending OMERO.server*.

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.
7.7 Setting the OMERO_HOME environment variable

**Warning:** OMERO_HOME should be considered strictly *internal* to OMERO and be reserved for development or “power usage”.

### 7.7.1 Rationale

OMERO_HOME defines the root deployment directory to be used by OMERO command line tools. Setting it allows to switch between different versions of applications or libraries during development.

*Note:* Despite a familiar-sounding name, OMERO_HOME does not denote the server installation directory.

### 7.7.2 Caveats

As a “power switch”, OMERO_HOME will override the directory layout as detected by the *bin/omero* family of utilities. This could cause hard-to-detect problems at runtime. Environments hosting multiple OMERO versions would be particularly prone to such side effects, as illustrated below.

```bash
# Installation directory layout:
# /opt
#   +++ OMERO.server-version-4.4.10
#   +++ OMERO.server-version-5.0.0

# OMERO-4 was first installed, and is conveniently referenced through OMERO_HOME
$ echo $OMERO_HOME
> /opt/OMERO.server-version-4.4.10

# Update the configuration for OMERO-4
$ /opt/OMERO.server-version-4.4.10/bin/omero config set omero.db.name 'omero_4_database'
$ /opt/OMERO.server-version-4.4.10/bin/omero config get omero.db.name
> omero_4_database

# Update the configuration for OMERO-5
$ /opt/OMERO.server-version-5.0.0/bin/omero config set omero.db.name 'omero_5_database'
$ /opt/OMERO.server-version-5.0.0/bin/omero config get omero.db.name
> omero_5_database

# Double check the OMERO-4 configuration...
# Using OMERO_HOME has taken precedence over the current directory
# The OMERO-4 server instance has been mistakenly updated to point at
# a version 5 database schema.
# Furthermore, the issue might go unnoticed until the next OMERO-4 restart.
$ /opt/OMERO.server-version-4.4.10/bin/omero config get omero.db.name
> omero_5_database
```

### 7.7.3 Recommendations

- If you are using an environment variable to define the server installation root, choose a different name, as represented by OMERO_PREFIX in this documentation.

- If you are nevertheless using OMERO_HOME in your *local* environment, it is worth checking for potentially lingering global definitions (e.g. *.bashrc*, *.profile*, /etc/profile.d/).

See also: [OMERO.server installation](#)
7.8 Configuration properties glossary

- Introduction
- Mandatory properties
- Binary repository
- Client
- Database
- Grid
- Ice
- JVM
- LDAP
- Mail
- Metrics
- Performance
- Pixeldata
- Policy
- Ports
- Scripts
- Search
- Security
- Web

7.8.1 Introduction

The primary form of configuration is via the use of key/value properties, stored in etc/grid/config.xml and read on server startup. Backing up and copying these properties is as easy as copying this file to a new server version.

The etc/omero.properties\(^{19}\) file of your distribution defines all the default configuration properties used by the server. Changes made to the file are not recognized by the server. Instead, configuration options can be set using the omero config set command:

$ bin/omero config set <parameter> <value>

When supplying a value with spaces or multiple elements, use single quotes. The quotes will not be saved as part of the value (see below).

To remove a configuration option (to return to default values where mentioned), simply omit the value:

$ bin/omero config set <parameter>

These options will be stored in a file: etc/grid/config.xml which you can read for reference. DO NOT edit this file directly.

You can also review all your settings by using:

$ bin/omero config get

which should return values without quotation marks.

A final useful option of omero config edit is:

\(^{19}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/etc/omero.properties
$ bin/omero config edit

which will allow for editing the configuration in a system-default text editor.

Note: Please use the escape sequence \" for nesting double quotes (e.g. \"["foo","bar\"]\") or wrap with ’ (e.g. ’["foo", "bar"]’).

Examples of doing this are on the main Unix and Windows pages, as well as the LDAP installation page.

7.8.2 Mandatory properties

The following properties need to be correctly set for all installations of the OMERO.server. Depending on your set-up, default values may be sufficient.

- omero.data.dir
- omero.db.host
- omero.db.name
- omero.db.pass

7.8.3 Binary repository

omero.checksum.supported

Checksum algorithms supported by the server for new file uploads, being any comma-separated non-empty subset of:

- Adler-32
- CRC-32
- MD5-128
- Murmur3-32
- Murmur3-128
- SHA1-160
- File-Size-64

In negotiation with clients, this list is interpreted as being in descending order of preference.

Default: SHA1-160, MD5-128, Murmur3-128, Murmur3-32, CRC-32, Adler-32, File-Size-64

omero.data.dir

Default: /OMERO/

omero.fs.repo.path

Template for FS managed repository paths. Allowable elements are:

%user% bob
%userId% 4
%group% bobLab
%groupId% 3
%year% 2011
%month% 01
%monthname% January
These are described further at FS configuration options

The path must be unique per fileset to prevent upload conflicts, which is why %time% includes milliseconds.

A // may be used as a path separator: the directories preceding it are created with root ownership, the remainder are the user’s. At least one user-owned directory must be included in the path.

The template path is created below omero.managed.dir, e.g. /OMERO/ManagedRepository/$omero.fs.repo.path/

Default: %user%_%userId%/%year%-%month%/%day%/%time%

**omero.fs.repo.path_rules**

Rules to apply to judge the acceptability of FS paths for writing into omero.managed.dir, being any comma-separated non-empty subset of:

- Windows required
- Windows optional
- UNIX required
- UNIX optional
- local required
- local optional

Minimally, the “required” appropriate for the server is recommended. Also applying “optional” rules may make sysadmin tasks easier, but may be more burdensome for users who name their files oddly. “local” means “Windows” or “UNIX” depending on the local platform, the latter being applied for Linux and Mac OS X.

Default: Windows required, UNIX required

**omero.managed.dir**

Default: ${omero.data.dir}/ManagedRepository

### 7.8.4 Client

**omero.client.download_as.max_size**

Clients disable download as jpg/png/tiff above max pixel count.

Default: 144000000

**omero.client.scripts_to_ignore**

Server-side scripts used in IScript service Clients shouldn’t display.
Default: /omero/figure_scripts/Movie_Figure.py, /omero/figure_scripts/Split_View_Figure.py, /omero/figure_scripts/Thumbnail_Figure.py, /omero/figure_scripts/ROI_Split_Figure.py, /omero/export_scripts/Make_Movie.py, /omero/setup_scripts/FLIM_initialise.py, /omero/import_scripts/Populate_ROI.py

**omero.client.ui.menu.dropdown.colleagues.enabled**

Flag to show/hide colleagues

Default: *true*

**omero.client.ui.menu.dropdown.colleagues.label**

Client dropdown menu colleagues label.

Default: ${omero.client.ui.menu.dropdown.colleagues}

**omero.client.ui.menu.dropdown.everyone.enabled**

Flag to show/hide all users.

Default: *true*

**omero.client.ui.menu.dropdown.everyone.label**

Client dropdown menu all users label.

Default: ${omero.client.ui.menu.dropdown.everyone}

**omero.client.ui.menu.dropdown.leaders.enabled**

Flag to show/hide leader.

Default: *true*

**omero.client.ui.menu.dropdown.leaders.label**

Client dropdown menu leader label.

Default: ${omero.client.ui.menu.dropdown.leaders}

**omero.client.ui.tree.orphans.description**

Description of the “Orphaned images” container.

Default: This is a virtual container with orphaned images. These images are not linked anywhere. Just drag them to the selected container.

**omero.client.ui.tree.orphans.enabled**

Flag to show/hide “Orphaned images” container. Only accept “true” or “false”

Default: *true*

**omero.client.ui.tree.orphans.name**

Name of the “Orphaned images” container located in client tree data manager.

Default: *Orphaned Images*
**omero.client.viewer.initial_zoom_level**

Initial client image viewer zoom level for big images

Default: 0

**omero.client.viewer.interpolate_pixels**

Client viewers interpolate pixels by default.

Default: true

**omero.client.viewer.roi_limit**

Client viewers roi limit.

Default: 2000

**omero.client.web.host**

Absolute omeroweb host http(s)://your_domain/prefix/

Default: [empty]

### 7.8.5 Database

**omero.db.authority**

The string that will be used as the base for LSIDs in all exported OME objects including OME-XML and OME-TIFF. It’s usually not necessary to modify this value since the database UUID (stored in the database) is sufficient to uniquely identify the source.

Default: export.openmicroscopy.org

**omero.db.dialect**

Implementation of the org.hibernate.dialect.Dialect interface which will be used to convert HQL queries and save operations into SQL SELECTs and DML statements.

(PostgreSQL default)

Default: ome.util.PostgresqlDialect

**omero.db.driver**

JDBC driver used to access the database. Other drivers can be configured which wrap this driver to provide logging, monitoring, etc.

(PostgreSQL default)

Default: org.postgresql.Driver

**omero.db.host**

The host name of the machine on which the database server is running. A TCP port must be accessible from the server on which OMERO is running.

Default: localhost
**omero.db.name**

The name of the database instance to which OMERO will connect.
Default: *omero*

**omero.db.pass**

The password to use to connect to the database server
Default: *omero*

**omero.db.patch**

The patch version of the database which is in use. This value need not match the patch version of the server that is being used with. Any changes by developers to the database schema will result in a bump to this value.
Default: 0

**omero.db.poolsize**

Sets the number of database server connections which will be used by OMERO. Your database installation will need to be configured to accept at least as many, preferably more, connections as this value.
Default: 10

**omero.db.port**

TCP port on which the database server is listening for connections. Used by the JDBC driver to access the database. Use of a local UNIX socket is not supported.
(PostgreSQL default)
Default: 5432

**omero.db.prepared_statement_cache_size**

Default: 10

**omero.db.profile**

Default values for the current profile will be hard-coded into the hibernate.properties file in the *model-*.jar. By using a different jar, you can modify the defaults.

Note: some other properties are defined in the file *etc/profiles/*omero.db.profile* Especially of importance is *omero.db.port*

Default: *psql*

**omero.db.sql_action_class**

Implementation of the ome.util.SqlAction interface which will be used to perform all direct SQL actions, i.e. without Hibernate.
(PostgreSQL default)
Default: *ome.util.actions.PostgresSqlAction*
**omero.db.statistics**

Whether JMX statistics are collected for DB usage (by Hibernate, etc)

Default: `true`

**omero.db.user**

The username to use to connect to the database server

Default: `omero`

**omero.db.version**

Version of the database which is in use. This value typically matches the major.minor version of the server that it is being used with. Typically, only developers will change this version to bump to a new major version.

Default: `OMERO5.2`

### 7.8.6 Grid

**omero.cluster.read_only**

Default: `false`

**omero.cluster.redirector**

Default: `nullRedirector`

**omero.grid.registry_timeout**

registry_timeout is the milliseconds which the registry and other services will wait on remote services to respond.

Default: `5000`

### 7.8.7 Ice

**Ice.IPv6**

Disable IPv6 by setting to 0. Only needed in certain situations.

Default: `1`

### 7.8.8 JVM

**omero.jvmcfg.append**

Contains other parameters which should be passed to the JVM. The value of “append” is treated as if it were on the command-line and so will be separated on whitespace. For example, `'-XX:-PrintGC -XX:+UseCompressedOops'` would results in two new arguments.

Default: `[empty]`
**omero.jvmcfg.heap_dump**
Toggles on or off heap dumps on OOMs. Default is “off”. The special value “tmp” will create the heap dumps in your temp directory.
Default: [empty]

**omero.jvmcfg.heap_size**
Explicit value for the -Xmx argument, e.g. “1g”
Default: [empty]

**omero.jvmcfg.max_system_memory**
Suggestion for strategies as to the maximum memory that they will use for calculating JVM settings (MB).
Default: 48000

**omero.jvmcfg.min_system_memory**
Suggestion for strategies as to the minimum memory that they will use for calculating JVM settings (MB).
Default: 3414

**omero.jvmcfg.percent**
Used only by the percent strategy. An integer between 0 and 100 which is the percent of active memory that will be used by the service.
Default: [empty]

**omero.jvmcfg.perm_gen**
Explicit value for the MaxPermSize argument to the JVM, e.g. “500M”. Ignored for Java8+
Default: [empty]

**omero.jvmcfg.strategy**
Memory strategy which will be used by default. Options include: percent, manual
Default: percent

**omero.jvmcfg.system_memory**
Manual override of the total system memory that OMERO will think is present on the local OS (MB). If unset, an attempt will be made to detect the actual amount: first by using the Python library psutil and if that is not installed, by running a Java tool. If neither works, 4.0GB is assumed.
Default: [empty]

### 7.8.9 LDAP

**omero.ldap.base**
LDAP server base search DN, i.e. the filter that is applied to all users. (can be empty in which case any LDAP user is valid)
Default: ou=example, o=com
Enable or disable LDAP (true or false).
Default: false

Default: (objectClass=groupOfNames)

Default: name=cn

Without a prefix the "new_user_group" property specifies the name of a single group which all new users will be added to. Other new_user_group strings are prefixed with :x: and specify various lookups which should take place to find one or more target groups for the new user.

:ou: uses the final organizational unit of a user’s dn as the single OMERO group e.g. omero.ldap.new_user_group=:ou:

:attribute: uses all the values of the specified attribute as the name of multiple OMERO groups. e.g. omero.ldap.new_user_group=:attribute:memberOf

Like :attribute:, :filtered_attribute: uses all the values of the specified attribute as the name of multiple OMERO groups but the attribute must pass the same filter as :query: does. e.g. omero.ldap.new_user_group=:filtered_attribute:memberOf

A combination of filtered_attribute and dn_attribute, :filtered_dn_attribute: uses all of the values of the specified attribute as the DN of multiple OMERO groups but the attribute must pass the same filter as :query: e.g. omero.ldap.new_user_group=:filtered_dn_attribute:memberOf

:query: performs a query for groups. The “name” property will be taken as defined by omero.ldap.group_mapping and the resulting filter will be AND’ed with the value group_filter (above) e.g. omero.ldap.new_user_group=:query:(member=@{dn})

:bean: looks in the server’s context for a bean with the given name which implements ome.security.auth.NewUserGroupBean e.g. omero.ldap.new_user_group=:bean:myNewUserGroupMapperBean

Default: default

A query element to check if user who is being created via the new_user_group setting should be made a “manager”, i.e. owner, of the queried group. E.g. omero.ldap.new_user_group_owner=(owner=@{dn}) will use the ‘manager’ attribute to set the ‘owner’ flag in the database. This query element is appended to any query used by new_user_group with an AND.

This property is not used by new_user_group type ‘default’ and only potentially by :bean:.

Default: [empty]

LDAP server bind password (if required; can be empty)
Default: [empty]
**omero.ldap.referral**

Available referral options are: “ignore”, “follow”, or “throw” as per the JNDI referral documentation.

Default: **ignore**

**omero.ldap.sync_on_login**

Whether or not values from LDAP will be synchronized to OMERO on each login. This includes not just the username, email, etc, but also the groups that the user is a member of.

Note: Admin actions carried out in the clients may not survive this synchronization e.g. LDAP users removed from an LDAP group in the UI will be re-added to the group when logging in again after the synchronization.

Default: **false**

**omero.ldap.urls**

Set the URL of the LDAP server. A SSL URL for this property would be of the form: ldaps://ldap.example.com:636

Default: **ldap://localhost:389**

**omero.ldap.user_filter**

Default: **(objectClass=person)**

**omero.ldap.user_mapping**

Default: **omeName=cn, firstName=givenName, lastName=sn, email=mail, institution=department, middleName=middleName**

**omero.ldap.username**

LDAP server bind DN (if required; can be empty)

Default: **[empty]**

### 7.8.10 Mail

**omero.mail.bean**

Mail sender properties

Default: **defaultMailSender**

**omero.mail.config**

Enable or disable mail sender (**true** or **false**).

Default: **false**

**omero.mail.from**

the email address used for the “from” field

Default: **omero@$({omero.mail.host})**
**omero.mail.host**
the hostname of smtp server
Default: *localhost*

**omero.mail.password**
the password to connect to the smtp server (if required; can be empty)
Default: *[empty]*

**omero.mail.port**
the port of smtp server
Default: 25

**omero.mail.smtp.auth**
see javax.mail.Session properties
Default: *false*

**omero.mail.smtp.connectiontimeout**
Default: 60000

**omero.mail.smtp.debug**
Default: *false*

**omero.mail.smtp.socketFactory.class**
Default: *javax.net.SocketFactory*

**omero.mail.smtp.socketFactory.fallback**
Default: *false*

**omero.mail.smtp.socketFactory.port**
Default: *${omero.mail.port}*

**omero.mail.smtp.starttls.enable**
Default: *false*

**omero.mail.smtp.timeout**
Default: 60000
**omero.mail.transport.protocol**

other smtp parameters; see org.springframework.mail.javamail.JavaMailSenderImpl

Default: *smtp*

**omero.mail.username**

the username to connect to the smtp server (if required; can be empty)

Default: *empty*

### 7.8.11 Metrics

**omero.metrics.bean**

Which bean to use: nullMetrics does nothing defaultMetrics uses the properties defined below

Default: *defaultMetrics*

**omero.metrics.graphite**

Address for Metrics to send server data

Default: *empty*

**omero.metrics.slf4j_minutes**

Number of minutes to periodically print to slf4j 0 or lower disables the printout.

Default: 60

### 7.8.12 Performance

**omero.sessions.maximum**

Default: 0

**omero.sessions.sync_force**

Default: 1800000

**omero.sessions.sync_interval**

Default: 120000

**omero.sessions.timeout**

Sets the duration of inactivity in milliseconds after which a login is required.

Default: 600000

**omero.threads.cancel_timeout**

Default: 5000
omero.threads.idle_timeout
Default: 5000

omero.threads.max_threads
Default: 50

omero.threads.min_threads
Default: 5

omero.throttling.method_time.error
Time in milliseconds after which a single method invocation will print a ERROR statement to the server log. If ERRORs are frequently being printed to your logs, you may want to increase this value after checking that no actual problem exists. Values of more than 60000 (1 minute) are not advised.
Default: 20000

omero.throttling.method_time.error.indexer
Value for the indexer is extended to 1 day
Default: 86400000

omero.throttling.method_time.warn
Time in milliseconds after which a single method invocation will print a WARN statement to the server log.
Default: 5000

omero.throttling.method_time.warn.indexer
Value for the indexer is extended to 1 hour
Default: 3600000

omero.throttling.objects_read_interval
Default: 1000

omero.throttling.objects_written_interval
Default: 1000

omero.throttling.servants_per_session
Default: 10000
7.8.13 Pixeldata

**omero.pixeldata.backoff**

Name of the spring bean which will be used to calculate the backoff (in ms) that users should wait for an image to be ready to view.

Default: *ome.io.nio.SimpleBackOff*

**omero.pixeldata.batch**

Number of instances indexed per indexing. (Ignored by pixelDataEventLogQueue)

Default: *50*

**omero.pixeldata.cron**

Polling frequency of the pixeldata processing. Set empty to disable pixeldata processing.

Cron Format: seconds minutes hours day-of-month month day-of-week year (optional). For example, "0,30 * * * * ?" is equivalent to running every 30 seconds. For more information download the latest *1.x version* of the Quartz Job Scheduler[^1] and review docs/api/org/quartz/CronExpression.html within the distribution.

Default: */4 * * * * ?*

**omero.pixeldata.dispose**

Whether the PixelData.dispose() method should try to clean up ByteBuffer instances which may lead to memory exceptions. See ticket #11675 for more information. Note: the property is set globally for the JVM.

Default: *true*

**omero.pixeldata.event_log_loader**

EventLogLoader that will be used for loading EventLogs for the action “PIXELDATA”. Choices include: pixelDataEventLogQueue and the older pixelDataPersistentEventLogLoader

Default: *pixelDataEventLogQueue*

**omero.pixeldata.max_plane_height**

Default: *3192*

**omero.pixeldata.max_plane_width**

Default: *3192*

**omero.pixeldata.memoizer_wait**

Maximum time in milliseconds that file parsing can take without the parsed metadata being cached to BioFormatsCache.

Default: *0*

**omero.pixeldata.repetitions**

Instead, it is possible to tell the server to run more pixeldata repetitions, each of which gets completely committed before the next. This will only occur when there is a substantial backlog of pixels to process.

(Ignored by pixelDataEventLogQueue; uses threads instead)

Default: 1

**omero.pixeldata.threads**

How many pixel pyramids will be generated at a single time. The value should typically not be set to higher than the number of cores on the server machine.

Default: 2

**omero.pixeldata.tile_height**

Default: 256

**omero.pixeldata.tile_sizes_bean**

Default sizes for tiles are provided by a ome.io.nio.TileSizes implementation. By default the bean ("configuredTileSizes") uses the properties provided here.

Default: configuredTileSizes

**omero.pixeldata.tile_width**

Default: 256

### 7.8.14 Policy

**omero.policy.bean**

Instance of the PolicyService interface which will be responsible for checking certain server actions made by a user.

Default: defaultPolicyService

**omero.policy.binary_access**

Configuration for the policy of whether users can access binary files from disk. Binary access includes all attempts to download a file from the UI.

The individual components of the string include:

- write - whether or not users who have WRITE access to the objects can access the binary. This includes group and system administrators.
- read - whether or not users who have READ access to the objects can access the binary.
- image - whether or not images are to be considered accessible as a rule.
- plate - whether or not plates and contained HCS objects are to be considered accessible as a rule. This includes wells, well samples, and plate runs.

Though the order of the components of the property are not important, the order that they are listed above roughly corresponds to their priority. E.g. a -write value will override +plate.

Example 1: “-read,+write,+image,-plate” only owners of an image and admins can download it.

Example 2: “-read,-write,-image,-plate” no downloading is possible.
Configuration properties of the same name can be applied to individual groups as well. E.g. adding, `omero.policy.binary_access=-read` to a group, you can prevent group-members from downloading original files.

Configuration is pessimistic: if there is a negative either on the group or at the server-level, the restriction will be applied. A missing value at the server restricts the setting but allows the server to override.

Default: `+read, +write, +image`

### 7.8.15 Ports

**omero.ports.prefix**

The prefix to apply to all port numbers (SSL, TCP, registry) used by the server  
Default: `[empty]`

**omero.ports.registry**

The IceGrid registry port number to use  
Default: `4061`

**omero.ports.ssl**

The Glacier2 SSL port number to use  
Default: `4064`

**omero.ports.tcp**

The Glacier2 TCP port number to use  
Default: `4063`

### 7.8.16 Scripts

**omero.launcher.jython**

Executable on the PATH which will be used for scripts with the mimetype `text/x-jython`.  
Default: `jython`

**omero.launcher.matlab**

Executable on the PATH which will be used for scripts with the mimetype `text/x-matlab`.  
Default: `matlab`

**omero.launcher.python**

Executable on the PATH which will be used for scripts with the mimetype `text/x-python`.  
No value implies use `sys.executable`  
Default: `[empty]`
**omero.process.jython**

Server implementation which will be used for scripts with the mimetype 'text/x-jython'. Changing this value requires that the appropriate class has been installed on the server.

Default: `omero.processor.ProcessI`

**omero.process.matlab**

Server implementation which will be used for scripts with the mimetype 'text/x-matlab'. Changing this value requires that the appropriate class has been installed on the server.

Default: `omero.processor.MATLABProcessI`

**omero.process.python**

Server implementation which will be used for scripts with the mimetype 'text/x-python'. Changing this value requires that the appropriate class has been installed on the server.

Default: `omero.processor.ProcessI`

**omero.scripts.cache.cron**

Frequency to reload script params. By default, once a day at midnight.

Cron Format: seconds minutes hours day-of-month month day-of-week year (optional). For example, “0,30 * * * *?” is equivalent to running every 30 seconds. For more information download the latest 1.x version of the Quartz Job Scheduler and review `docs/api/org/quartz/CronExpression.html` within the distribution.

Default: `0 0 0 * * ?`

**omero.scripts.cache.spec**

Guava LoadingCache spec for configuring how many script JobParams will be kept in memory for how long.

For more information, see [http://google.github.io/guava/releases/17.0/api/docs/com/google/common/cache/CacheBuilderSpec.html](http://google.github.io/guava/releases/17.0/api/docs/com/google/common/cache/CacheBuilderSpec.html)

Default: `maximumSize=1000`

**omero.scripts.timeout**

Default: `3600000`

### 7.8.17 Search

**omero.search.analyzer**

Analyzer used both index and to parse queries

Default: `ome.services.fulltext.FullTextAnalyzer`

**omero.search.batch**

Size of the batches to process events per indexing. Larger batches can speed up indexing, but at the cost of memory.

Default: `5000`

---

oreomero.search.bridges

Extra bridge classes, comma-separated, to be invoked on each indexing. Bridges are used to parse more information out of the data.

Default: [empty]

oreomero.search.cron

Polling frequency of the indexing. Set empty to disable search indexing.

Cron Format: seconds minutes hours day-of-month month day-of-week year (optional). For example, “0,30 * * * * ?” is equivalent to running every 30 seconds. For more information download the latest 1.x version of the Quartz Job Scheduler\(^2\) and review docs/api/org/quartz/CronExpression.html within the distribution.

Default: */2 * * * * ?

oreomero.search.event_log_loader

Default: eventLogQueue

oreomero.search.excludes

Indexing takes place on all EventLogs as they occur in the database. The types listed here will be skipped if they appear in the “entityType” field of the EventLog table.


oreomero.search.include_actions

EventLog action values which will be indexed. Unless custom code is generating other action types, this property should not need to be modified.

Default: INSERT, UPDATE, REINDEX, DELETE

oreomero.search.include_types

Whitelist of object types which will be indexed. All other types will be ignored. This matches the currently available UI options but may need to be expanded for custom search bridges.


oreomero.search.locking_strategy

Default: native

\(^2\)http://www.quartz-scheduler.org/downloads/
**omero.search.max_file_size**

Maximum file size for text indexing (bytes). If a file larger than this is attached, e.g., to an image, the indexer will simply ignore the contents of the file when creating the search index. This should not be set to more than half of the Indexer heap space.

**Note:** If you set the max file size to greater than 1/2 the size of the indexer’s heap (256 MB by default), you may encounter Out of Memory errors in the Indexer process or you may cause the search index to become corrupt. Be sure that you also increase the heap size accordingly (see *OutOfMemoryError / PermGen space errors in OMERO.server logs*).

Default: 131072000

**omero.search.max_partition_size**

Number of objects to load in a single indexing window. The larger this value the fewer times a single object will be indexed unnecessarily. Each object uses roughly 100 bytes of memory.

Default: 1000000

**omero.search.maxclause**

Maximum number of OR-clauses to which a single search can expand

Default: 4096

**omero.search.merge_factor**

Default: 25

**omero.search.ram_buffer_size**

Default: 64

**omero.search.repetitions**

Instead, it is possible to tell the server to run more indexing repetitions, each of which gets completely committed before the next. This will only occur when there is a substantial backlog of searches to perform. (More than 1 hour's worth)

Default: 1

**omero.search.reporting_loops**

Periodically the completion percentage will be printed. The calculation can be expensive and so is not done frequently.

Default: 100

**7.8.18 Security**

**omero.security.chmod_strategy**

Default: groupChmodStrategy

**omero.security.filter.bitand**

Default: \((\text{int8 applications (permissions, %s)} = \text{%s})\)
**omero.security.keyStore**

A keystore is a database of private keys and their associated X.509 certificate chains authenticating the corresponding public keys. A keystore is mostly needed if you are doing client-side certificates for authentication against your LDAP server.

Default: [empty]

**omero.security.keyStorePassword**

Sets the password of the keystore

Default: [empty]

**omero.security.login_failure_throttle_count**

Default: 1

**omero.security.login_failure_throttle_time**

Default: 3000

**omero.security.password_provider**

Implementation of PasswordProvider that will be used to authenticate users. Typically, a chained password provider will be used so that if one form of authentication (e.g. LDAP) does not work, other attempts will be made.

Default: chainedPasswordProvider

**omero.security.password_required**

Controls whether the server will allow creation of user accounts with an empty password. If set to true (default, strict mode), empty passwords are disallowed. This still allows the guest user to interact with the server.

Default: true

**omero.security.trustStore**

A truststore is a database of trusted entities and their associated X.509 certificate chains authenticating the corresponding public keys. The truststore contains the Certificate Authority (CA) certificates and the certificate(s) of the other party to which this entity intends to send encrypted (confidential) data. This file must contain the public key certificates of the CA and the client’s public key certificate.

Default: [empty]

**omero.security.trustStorePassword**

Sets the password of the truststore

Default: [empty]

### 7.8.19 Web

**omero.webadmins**

A list of people who get code error notifications whenever the application identifies a broken link or raises an unhandled exception that results in an internal server error. This gives the administrators immediate notification of any errors, see OMERO.mail.

Example: ‘["Full Name", "email address"]’

Default: []
**OMERO.web.application_server**

OMERO.web is configured to run in Gunicorn as a generic WSGI application by default. If you are using Apache change this to "wsgi" before generating your web server configuration. Available options: “wsgi-tcp” (Gunicorn), “wsgi” (Apache)
Default: *wsgi-tcp*

**omero.web.application_server.host**

Upstream application host
Default: *127.0.0.1*

**omero.web.application_server.max_requests**

The maximum number of requests a worker will process before restarting.
Default: *0*

**omero.web.application_server.port**

Upstream application port
Default: *4080*

**omero.web.apps**

Add additional Django applications. For example, see *Creating an app*
Default: *

**omero.web.caches**

OMERO.web offers alternative session backends to automatically delete stale data using the cache session store backend, see Django cached session documentation for more details.
Default: *

**omero.web.chunk_size**

Size, in bytes, of the “chunk”
Default: *1048576*

**omero.web.databases**

Default: *

**omero.web.debug**

A boolean that turns on/off debug mode.
Default: *false*

---

23https://docs.djangoproject.com/en/1.8/topics/http/sessions/#using-cached-sessions
**omero.web.index_template**

Define template used as an index page `http://your_host/omero/`. If None user is automatically redirected to the login page. For example use ‘webclient/index.html’.

Default: `None`

**omero.web.logdir**

A path to the custom log directory.

Default: `/home/omero/OMERO.server/var/log`

**omero.web.login_logo**

Customize webclient login page with your own logo. Logo images should ideally be 150 pixels high or less and will appear above the OMERO logo. You will need to host the image somewhere else and link to it with "http://www.openmicroscopy.org/site/logo.jpg".

Default: `None`

**omero.web.login_redirect**

Redirect to the given location after logging in. It only supports arguments for Django reverse function\(^24\). For example: `{"redirect": ["webindex"], "viewname": "load_template", "args": ["userdata"], "query_string": "experimenter=-1"}`

Default: `{}`

**omero.web.login_view**

Default: `weblogin`

**omero.web.page_size**

Number of images displayed within a dataset or ‘orphaned’ container to prevent from loading them all at once.

Default: `200`

**omero.web.ping_interval**

Timeout interval between ping invocations in seconds

Default: `60000`

**omero.web.pipeline_css_compressor**

Compressor class to be applied to CSS files. If empty or None, CSS files won’t be compressed.

Default: `None`

**omero.web.pipeline_js_compressor**

Compressor class to be applied to JavaScript files. If empty or None, JavaScript files won’t be compressed.

Default: `None`

\(^24\)https://docs.djangoproject.com/en/1.8/ref/urlresolvers/#django.core.urlresolvers.reverse

---

**7.8. Configuration properties glossary**

256
**omero.web.pipeline_staticfile_storage**

The file storage engine to use when collecting static files with the collectstatic management command. See the documentation[^1] for more details.

Default: `pipeline.storage.PipelineStorage`

**omero.web.prefix**

Used as the value of the SCRIPT_NAME environment variable in any HTTP request.

Default: `None`

**omero.web.public.cache.enabled**

Default: `false`

**omero.web.public.cache.key**

Default: `omero.web.public.cache.key`

**omero.web.public.cache.timeout**

Default: `86400`

**omero.web.public.enabled**

Enable and disable the OMERO.web public user functionality.

Default: `false`

**omero.web.public.password**

Password to use during authentication.

Default: `None`

**omero.web.public.server_id**

Server to authenticate against.

Default: `1`

**omero.web.public.url_filter**

Set a URL filter for which the OMERO.web public user is allowed to navigate. The idea is that you can create the public pages yourself (see OMERO.web framework since we do not provide public pages).

Default: `^/(?!webadmin)`

**omero.web.public.user**

Username to use during authentication.

Default: `None`

**omero.web.secret_key**

A boolean that sets SECRET_KEY for a particular Django installation.

Default: None

**omero.web.secure_proxy_ssl_header**

A tuple representing a HTTP header/value combination that signifies a request is secure. Example ‘[“HTTP_X_FORWARDED_PROTO_OMERO_WEB”, “https”]’. For more details see secure proxy ssl header.

Default: []

**omero.web.server_list**

A list of servers the Web client can connect to.

Default: [“localhost”, 4064, “omero”]

**omero.web.session_cookie_age**

The age of session cookies, in seconds.

Default: 86400

**omero.web.session_cookie_domain**

The domain to use for session cookies

Default: None

**omero.web.session_cookie_name**

The name to use for session cookies

Default: None

**omero.web.session_engine**

Controls where Django stores session data. See Configuring the session engine for more details.

Default: omeroweb.filesessionstore

**omero.web.session_expire_at_browser_close**

A boolean that determines whether to expire the session when the user closes their browser. See Django Browser-length sessions vs. persistent sessions documentation for more details.

Default: true

**omero.web.static_root**

The absolute path to the directory where collectstatic will collect static files for deployment. If the staticfiles contrib app is enabled (default) the collectstatic management command will collect static files into this directory.

Default: /home/omero/OMERO.server/lib/python/omeroweb/static

---

26 https://docs.djangoproject.com/en/1.8/ref/settings/#secure-proxy-ssl-header
27 https://docs.djangoproject.com/en/1.8/ref/settings/#session-engine
28 https://docs.djangoproject.com/en/1.8/topics/http/sessions/#browser-length-vs-persistent-sessions
omero.web.static_url

URL to use when referring to static files. Example: `/static/` or `http://static.example.com/`. Used as the base path for asset definitions (the Media class) and the staticfiles app. It must end in a slash if set to a non-empty value.
Default: `/static/`

omero.web.staticfile_dirs

Defines the additional locations the staticfiles app will traverse if the FileSystemFinder finder is enabled, e.g. if you use the collectstatic or findstatic management command or use the static file serving view.
Default: `[]`

omero.web.template_dirs

List of locations of the template source files, in search order. Note that these paths should use Unix-style forward slashes, even on Windows.
Default: `[]`

omero.web.ui.center_plugins

Add plugins to the center panels. Plugins are `['Channel overlay', 'webtest/webclient_plugins/center_plugin.overlay.js.html', 'channel_overlay_panel']`. The javascript loads data into `$('#div_id')`.
Default: `[]`

omero.web.ui.right_plugins

Add plugins to the right-hand panel. Plugins are `['Label', 'include.js', 'div_id']`. The javascript loads data into `$('#div_id')`.
Default: `[["Acquisition", "webclient/data/includes/right_plugin.acquisition.js.html", "metadata_tab"], ["Preview", "webclient/data/includes/right_plugin.preview.js.html", "preview_tab"]]

omero.web.ui.top_links

Add links to the top header: links are `['Link Text', 'link', options]`, where the url is reverse('link') OR simply 'link' (for external urls). E.g. `[['Webtest", "webtest_index"], ["Homepage", "http://...", {"title": "Homepage", "target": "new"} ]]`

omero.web.use_x_forwarded_host

Specifies whether to use the X-Forwarded-Host header in preference to the Host header. This should only be enabled if a proxy which sets this header is in use.
Default: `false`

omero.web.viewer.view

Django view which handles display of, or redirection to, the desired full image viewer.
Default: `omeroweb.webclient.views.image_viewer`
**OMERO Documentation, Release 5.2.6**

**omero.web.webgateway_cache**

Default: *None*

**omero.web.wsgi_args**


Default: *None*

**omero.web.wsgi_timeout**

Workers silent for more than this many seconds are killed and restarted. Check Gunicorn Documentation [http://docs.gunicorn.org/en/stable/settings.html#timeout](http://docs.gunicorn.org/en/stable/settings.html#timeout)

Default: *60*

**omero.web.wsgi_workers**


Default: *5*

---

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

---

**7.9 Syslog configuration**

Syslog is a standard for message logging over networks. OMERO.server supports logging to either a local or remote syslog service.

This allows all logs of the OMERO.server to be routed to a central location instead of (or as well as) to a file.

**Note:** It is important to note that this applies only to the OMERO.server itself, not to components like OMERO.web.

---

**7.9.1 How it works**

Whenever a log message is generated, OMERO’s logging framework will forward that message to any configured appenders.

By default, OMERO is configured to log everything to files.

**Note:** OMERO is configured to log a record of events for operations such as import. These are written directly to the Managed Repository. It is very likely that even if replacing file logging with syslog, this aspect should be retained in files. This is easily achieved by not changing any loggers using SIFT.

---

**Configuration**

To configure OMERO to be able to log to syslog, it is necessary to modify the file OMERO.server/current/etc/logback.xml. It is possible to do all the configuration changes in this file alone, but for ease of config management, it is demonstrated here where an additional OMERO.server/current/etc/logback_syslog.xml file is used in addition.

The following information is required to configure OMERO to log to syslog.

---

https://en.wikipedia.org/wiki/Syslog
- The host on which syslog is running: e.g. localhost
- The port number on which syslog is running on that host: e.g. 514
- The facility (RFC 3164) that OMERo should be handled as: e.g. user or local6

**Note:** The facility is important because it determines how syslog will handle the messages it receives. It is unlikely that OMERo’s log output will be desired in a local system’s primary message log for example. On Linux this is often /var/log/messages. Remember to configure the syslog configuration to avoid this. This is also where configuration of onward forwarding can be configured (to a service such as splunk). Finally, syslog can be configured to specifically output this facility output to a file such as /var/log/omero.

Create the new file OMERo.server/current/etc/logback_syslog.xml:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<included>
  <!-- syslog -->
  <appender name="SYSLOG" class="ch.qos.logback.classic.net.SyslogAppender">
    <!-- Exclude debug level logging from ome.services.blitz.repo.ManagedImportRequestI -->
    <filter class="ch.qos.logback.core.filter.EvaluatorFilter">
      <evaluator>return Level.DEBUG.equals(Level.toLevel(level)) \&\& logger.equals("ome.services.blitz.repo.ManagedImportRequestI");</evaluator>
      <OnMismatch>NEUTRAL</OnMismatch>
      <OnMatch>DENY</OnMatch>
    </filter>
    <syslogHost>localhost</syslogHost>
    <facility>local6</facility>
    <suffixPattern>OMERO [%level] [%thread] %logger %msg</suffixPattern>
  </appender>
  
  <!-- Exclude debug level logging from omero.* (except allow omero.cmd.*) -->
  <filter class="ch.qos.logback.core.filter.EvaluatorFilter">
    <evaluator>return Level.DEBUG.equals(Level.toLevel(level)) \&\& logger.startsWith("omero.") \&\& !logger.startsWith("omero.cmd.");</evaluator>
    <OnMismatch>NEUTRAL</OnMismatch>
    <OnMatch>DENY</OnMatch>
  </filter>
</included>
```

This creates an appender that sends messages to syslog. `syslogHost` is the host on which syslog is running. No port is specified as 514 is the default. The `suffixPattern` is customizable. In this instance it is identical to OMERo’s file logger except an added “OMERO” identifier has been added for clarity. The name of the appender has been set to SYSLOG. The filters replicate the same behaviour from the default FILE appender.

**Note:** If configuring the appender directly in the OMERo.server/current/etc/logback.xml file, then the `included` tag should not be used.

Within the configuration tag of OMERo.server/current/etc/logback.xml add:

```xml
<include file="/path/to/OMERO.server/etc/logback_syslog.xml"/>
```

**Note:** The included file path can be relative, but note that it is NOT relative to the OMERo.server/current/etc/logback.xml file, but to the current directory set by OMERo. It is highly recommended to use a full path.

---

Finally, also within `OMERO.server/current/etc/logback.xml` modify the `root` tag to include a second `appender-ref` (It can also be replaced if the file logs are not desired or syslog will handle writing those to a file on OMERO’s behalf):

```xml
<root level="OFF">
    <appender-ref ref="SYSLOG"/>
    <appender-ref ref="FILE"/>
</root>
```

**Note:** A restart of OMERO will be necessary before this takes effect.
This chapter explains how to customize the appearance and functionality of OMERO clients to host images for groups or public viewing.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

8.1 Customize OMERO clients

The OMERO clients offer a flexible user interface that can be customized. The sections below describe how to set up these features.

Note that depending on the deployment choice, OMERO.web will not activate configuration changes until either gunicorn is restarted using `bin/omero web restart (nginx)` or the Apache webserver has been reloaded.

8.1.1 Index page

Create new custom template in `/your/path/to/templates/mytemplate/index.html` and add the following:

```bash
$ bin/omero config set omero.web.template_dirs '/your/path/to/templates/'
$ bin/omero config set omero.web.index_template 'mytemplate/index.html'
```

Note that users will no longer be automatically redirected to the login page once an index page is added.

8.1.2 Login page

`omero.web.login_logo` allows you to customize the webclient login page with your own logo. Logo images should ideally be 150 pixels high or less and will appear above the OMERO logo. You will need to host the image somewhere else and link to it with:

```bash
$ bin/omero config set omero.web.login_logo 'http://www.url/to/image.png'
```

8.1.3 Login redirection

`omero.web.login_redirect` property redirects to the given location after logging in:

```bash
$ bin/omero config set omero.web.login_redirect '{"redirect": ["webindex"], "viewname": "load_template", "query_string": "experimenter=-1"}'}
```
8.1.4 Top links menu

`omero.web.ui.top_links` adds links to the top header:
8.1.5 Group and Users in dropdown menu

Customize the groups and users dropdown menu by changing the labels or hiding the entire list:

```
$ bin/omero config set omero.client.ui.menu.dropdown.leaders "Owners"
$ bin/omero config set omero.client.ui.menu.dropdown.colleagues.enabled true
$ bin/omero config set omero.client.ui.menu.dropdown.colleagues "Members"
$ bin/omero config set omero.client.ui.menu.dropdown.colleagues.enabled true
$ bin/omero config set omero.client.ui.menu.dropdown.all "All Members"
$ bin/omero config set omero.client.ui.menu.dropdown.colleagues.enabled true
```

8.1.6 Orphaned container

`omero.client.ui.tree.orphans.name` allows you to change the name of the “Orphaned images” container located in the client data manager tree:
$ bin/omero config set omero.client.ui.tree.orphans.name "Orphaned images"

8.1.7 Disabling scripts

`omero.client.scripts_to_ignore` hides the scripts that the clients should not display:

$ bin/omero config append omero.client.scripts_to_ignore "/my_scripts/script.py"

8.1.8 Download restrictions

`omero.policy.binary_access` determines whether users can access binary files from disk. Binary access includes all attempts to download a file from the UI:

$ bin/omero config set omero.policy.binary_access "+read,+write,+image"

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.
8.2 Public data in the repository

The OMERO.web framework allows raw data to be published using built-in tools or supplied through webservices to external web pages. Selected datasets can be made visible to a ‘public user’ using the standard OMERO permissions system, ensuring you always have control over how users can interact with your data.

There are several ways of publishing data using OMERO.web:

- using a URL to launch the web-based Image viewer, as described in Launching OMERO.web viewer, which can be accompanied by a thumbnail. For more details of how to load the thumbnail see URLs from within OMERO.web
- embedding the image viewport directly into other web pages, for more details see Customizing the content of the embedded OMERO.web viewport
- allowing public access to the OMERO.web data manager
- writing your own app to host your public data (see Creating an app) and then allowing public access to the chosen URL for that app

The sections below describe how to set up these features.

8.2.1 Public user

The OMERO.web framework supports auto-login for a single username / password. This means that any public visitors to certain OMERO.web pages will be automatically logged in and will be able to access the data available to the defined ‘public user’.

To set this up on your OMERO.web installation:

- Create a group with read-only permissions (the name can be anything e.g. “public-data”). We recommend read-only permissions so that the public user will not be able to modify, delete or annotate data belonging to other members.
- Create a member of this group, noting the username and password (you will enter these below). Again, the First Name, Last Name, username and password can be anything you like.

**Note:** If you add this member to other groups, all data in these groups will also become publicly accessible for as long as this user remains in the group.

- Enable the omero.web.public.enabled property and set omero.web.public.user and omero.web.public.password:
Set the `omero.web.public.url_filter`. This filter is a regular expression that will allow only matching URLs to be accessed by the public user.

There are three common use cases for the URL filter:

- Enable ‘webgateway’ URLs which includes everything needed for the full image viewer:
  ```sh
  $ bin/omero config set omero.web.public.url_filter '^/webgateway'
  ```

  without the ability to download data:
  ```sh
  $ bin/omero config set omero.web.public.url_filter '^/webgateway/(?!archived_files|download_as)'
  ```

  Then you can access public images via the following link `http://your_host/webgateway/img_detail/IMAGE_ID/`.

- Create your own public pages in a separate app (see `create app`) and allow public access to that app. For example, to allow only URLs that start with ‘/my_web_public’ you would use:
  ```sh
  $ bin/omero config set omero.web.public.url_filter '/my_web_public'
  ```

- You can use the full webclient UI for public browsing of images. However, the webclient UI was not designed for public use and allows various actions that create data or are resource intensive. These can be selectively disabled using the following command:
  ```sh
  $ bin/omero config set omero.web.public.url_filter '/(webadmin/myphoto/|webclient/(?!act:on|logout|annotate_(file|tags|comment|rating|map)|script_ui|ome_tiff|figure_script)|webgateway/(?!archived_files|download_as))'
  ```

Set the `omero.web.public.server_id` which the public user will be automatically connected to. Default: 1 (the first server in the `omero.web.server_list`)

```sh
$ bin/omero config set omero.web.public.server_id 1
```

If you enable public access to the main webclient but still wish registered users to be able to login, the login page can always be accessed using a link of the form `https://your_host/webclient/login/`.

### 8.2.2 Reusing OMERO session

As an alternative to granting permanent public access to the data, the OMERO web framework supports password-less, OMERO session key-based authentication. For example a direct link to image will look as follows:

`https://your_host/webgateway/img_detail/IMAGE_ID/?server=SERVER_ID&bsession=OMERO_SESSION_KEY`

**Note:** The `SERVER_ID` should match the index from the list set using `omero.web.server_list` from the server session you created. If your list contains only one server, the index will be 1.

For more details about how to create an OMERO session see `server-side session` or use the `command line interface` to create one.
This chapter contains details of how OMERO.fs allows you to import and store data with OMERO 5.

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

## 9.1 OMERO.dropbox

DropBox was originally designed as the first stage of the file system changes referred to as OMERO.fs. It utilizes a file system monitor to find newly uploaded files and run a fully automatic import on those files if possible. This release of OMERO.dropbox runs on the same machine as the OMERO.server and watches designated areas of the local filesystem for new or modified files. If those files are importable, then an automatic import is initiated. OMERO.dropbox is started automatically when the OMERO.server starts and it will run if the prerequisites below are met.

### 9.1.1 Prerequisites

In addition to the general **System requirements** OMERO.dropbox has the following more specific requirements:

- OMERO.dropbox is built on underlying OS file-notification system, and so is only available for specific versions of certain operating systems. OMERO.dropbox has been tested on the following systems:
  - Linux with kernel 2.6.13 and higher.
  - Mac OS 10.6 and later.
  - Windows XP, Vista, 7, Server 2003, Server 2008 and Server 2008R2, but see note below.

- In addition some platforms require further Python packages to be available:
  - Mac OS systems that use a macports install of Python will need to have FSEvents available in the PYTHONPATH. This will require a path of the form `/System/Library/Frameworks/Python.framework/Versions/2.X/Extras/lib/python/PyObjC/` to be added, according to the version of Python used.
  - The filesystem which OMERO.dropbox watches must be local to the given operating system. Watching a network-attached share (NAS) is strictly **not** supported.

**Note:** It is likely, but not guaranteed, that DropBox will run on later versions of Windows, however only tested versions are supported by default. Setting the property `omero.fs.platformCheck` to False in `etc/grid/templates.xml` or using `omero config set` and restarting the server will allow DropBox to be started on an untested version of Windows.

If you do successfully run DropBox on an as yet untested version of Windows please do let us know via the forums and mailing lists.

---

1. [http://www.openmicroscopy.org/site/community/](http://www.openmicroscopy.org/site/community/)
9.1.2 Using DropBox

In its default configuration the monitored area of the file system is a DropBox subdirectory of the OmeroBinaryRepository directory. The system administrator should create DropBox and then under that a directory for each user, using their omero username. The ownership and permissions should be set so that a user can copy files into their DropBox directory:

/OMERO/DropBox/amy
    /emily
    /edgar
    /root
    /zak

Experimenters can add subdirectories under their named directory for convenience. Copying or moving a file of an importable file type into a named directory or nested subdirectory will initiate an automatic import of that file for that user. Multi-file formats will be imported after the last required file of a set is copied into the directory. Images and plates will be imported into the group the user was last logged into, with images placed into Orphaned images.

Acquisition systems can then be configured to drop a user’s images into a given DropBox.

Note:

- The DropBox system is designed for image files to be copied in at normal acquisition rates. Copying many files en masse may result in files failing to import.
- It is also intended as a write-once system. Modifying an image after it has been imported may result in that modified image also being imported depending on the operating system and how the image was modified.
- Once directories are created within DropBox or files are copied or moved into DropBox they should not be moved, renamed or otherwise changed. Images may be imported again or already imported images may become unreadable.

9.1.3 Permissions

Changing the permissions of a directory within DropBox may result in duplicate imports as a newly readable directory appears identical to a new directory. If directories need to be modified it is recommended that the DropBox system is stopped and then restarted around any changes, as below.

```sh
$ bin/omero admin ice server disable DropBox
$ bin/omero admin ice server stop DropBox
$ bin/omero admin ice server disable MonitorServer
$ bin/omero admin ice server stop MonitorServer

# make any directory changes

$ bin/omero admin ice server enable MonitorServer
$ bin/omero admin ice server enable DropBox
```

Note: Any new files copied into DropBox during this disabled period will not be detected and thus not imported.

9.1.4 Log files

The log files var/log/FileServer.log, var/log/MonitorServer.log and var/log/DropBox.log will indicate success or otherwise of start-up of the two components. Once running, var/log/MonitorServer.log will log file events seen within designated file areas and var/log/DropBox.log will log the progress of any file imports.

9.1.5 Unicode path and file names

If file or path names contain Unicode characters this can cause DropBox to fail. This can be remedied by the use of a sitecustomize.py or usercustomize.py file containing the following:
import sys
reload(sys)
sys.setdefaultencoding('utf-8')

For more details on using customization files in Python see: site — Site-specific configuration hook\(^2\). For more discussion on this issue within OMERO see the forum post: Dropbox halts on certain unicode characters\(^3\).

**Note:** If a customization file is used and the OMERO server is upgraded please ensure the file is still available to DropBox after the upgrade.

## 9.1.6 Advanced use

OMERO.dropbox can be configured in several ways through etc/grid/templates.xml. In its default configuration, as detailed above, it monitors the subdirectory DropBox of the OMERO data directory for all users.

A number of the properties in templates.xml accept a semi-colon separated list of values. This extended configuration allows a site to watch multiple directories, and configure each for a different user, a different type of file, etc. Any value missing from the configuration (e.g. `value="1;;2"`) will be replaced by the default value.

One example alternative configuration would be to watch specific directories for specific users. In the example below two directories are monitored, one for user amy and one for zak:

```
<property name="omero.fs.importUsers" value="amy;zak"/>
<property name="omero.fs.watchDir" value="/home/amy/myData;/home/zak/work/data"/>
```

The remaining properties have been left at their default values for both users.

To limit DropBox to import only files belonging to specific image types the following property can be set,

```
<property name="omero.fs.readers" value="/home/amy/my_readers.txt;"/>
```

Here only the image types listed in `my_readers.txt` will be imported for the user amy while the system-wide `readers.txt` will be used for zak.

For a full description of the properties see below.

### Properties

Each property takes the form of a single item or a semi-colon separated list of items. Where the item is a list, values within that list should be comma separated.

- **importUsers**
  
  The importUsers is either default or a list of OMERO user names. In the case of the value being default, the same configuration is applied to all users and each subsequent configuration setting should be a single value. In the case of this value being a list of users, each subsequent value should be a list of the same length as the number of users. The default value is default.

  ```xml
  <property name="omero.fs.importUsers" value="default"/>
  ```

- **watchDir**
  
  The absolute directory path of interest for each user. The default is empty.

\(^2\)https://docs.python.org/2.7/library/site.html  
\(^3\)https://www.openmicroscopy.org/community/viewtopic.php?f=4&t=7810&p=15910
• eventTypes
For automatic import Creation and Modification events are monitored. It is also possible to monitor Deletion events though these are not used by DropBox. The default is Creation, Modification.

    <property name="omero.fs.eventTypes" value="Creation,Modification"/>

• pathMode
By default existing and newly created subdirectories are monitored. It is possible to restrict monitoring to a single directory ("Flat"), only existing subdirectories ("Recurse"), or all subdirectories ("Follow"). For DropBox to function correctly the mode should be Follow. The default is Follow.

    <property name="omero.fs.pathMode" value="Follow"/>

• whitelist
A list of file extensions of interest. An empty list implies all file extensions are monitored. The default is an empty list.

    <property name="omero.fs.whitelist" value=""/>

• blacklist
A list of subdirectories to ignore. Not currently supported.

    <property name="omero.fs.blacklist" value=""/>

• timeout
This timeout in seconds is used by one-shot monitors. This property is not used by DropBox.

    property name="omero.fs.timeout" value="0.0"/>

• blockSize
The number of events that should be propagated to DropBox in one go. Zero implies all events possible. The default is zero.

    <property name="omero.fs.blockSize" value="0"/>

• ignoreSysFiles
If this is True events concerning system files, such as filenames beginning with a dot or default new folder names, are ignored. The exact events ignored will be OS-dependent. The default is True.

    <property name="omero.fs.ignoreSysFiles" value="True"/>

• ignoreDirEvents
If this is True then the creation and modification of subdirectories is not reported to DropBox. The default is True.

    <property name="omero.fs.ignoreDirEvents" value="True"/>
• **dirImportWait**
  The time in seconds that DropBox should wait after being notified of a file before starting an import on that file. This allows for companion files or filesets to be copied. If a new file is added to a fileset during this wait period DropBox begins waiting again. The default is 60 seconds.

  <property name="omero.fs.dirImportWait" value="60"/>

• **fileBatch**
  The number of files that can be copied in before processing the batch. In cases where there are large numbers of files in a typical file set it may be more efficient to set this value higher. The default is 10.

  <property name="omero.fs.fileBatch" value="10"/>

• **throttleImport**
  The time in seconds that DropBox should wait after initiating an import before initiating a second import. If imports are started too close together connection issues can arise. The default is 10 seconds.

  <property name="omero.fs.throttleImport" value="10"/>

• **readers**
  A file of readers. If this is a valid file then it is used to filter those events that are of interest. Only files corresponding to a reader in the file will be imported. The default is empty.

  <property name="omero.fs.readers" value=""/>

• **importArgs**
  A string of extra arguments supplied to the importer. This could include, for example, an email address to report failed imports to: --report --email test@example.com. The default is empty. For details on available extra arguments see Import images.

  <property name="omero.fs.importArgs" value=""/>

### Example

Here’s a full example of a configuration for two users:

```
<property name="omero.fs.importUsers" value="amy;zak"/>
<property name="omero.fs.watchDir" value="/home/amy/myData;/home/zak/work/data"/>
<property name="omero.fs.eventTypes" value="Creation,Modification;Creation,Modification"/>
<property name="omero.fs.pathMode" value="Follow;Follow"/>
<property name="omero.fs.whitelist" value=";"/>
<property name="omero.fs.blacklist" value=";"/>
<property name="omero.fs.timeout" value="0.0;0.0"/>
<property name="omero.fs.blockSize" value="0;0"/>
<property name="omero.fs.ignoreSysFiles" value="True;True"/>
<property name="omero.fs.ignoreDirEvents" value="True;True"/>
<property name="omero.fs.dirImportWait" value="60;60"/>
<property name="omero.fs.fileBatch" value="10;10"/>
<property name="omero.fs.throttleImport" value="10;10"/>
<property name="omero.fs.readers" value="/home/amy/my_readers.txt;"/>
<property name="omero.fs.importArgs" value="--report;--report --email zak@example.com"/>
```
9.2 In-place import

In-place import is a feature added to OMERO 5.0.0 to allow files which are already present on the server machine to be imported into OMERO without the need to copy them. This requires users to have shell (SSH, etc.) access to the server machine, and so there are a number of limitations to this implementation. Development of this feature is on-going, with improvements planned to enable a more user-friendly experience. This CLI-based stop-gap is being made available at this stage because for some users, in-place import is essential for their use of OMERO.

This feature is designed to allow imaging facilities to import large datasets into OMERO while keeping them safely stored in a secure repository which is read-only for users. Leaving the data in a user’s file system is very dangerous as they may forget they need to keep it or move to a different institution. Under no circumstances should in-place import be used with temporary storage.

**Warning:** The instructions below should help you get started but it is critical that you understand the implications of using this feature. Please do not just copy commands and hope for the best.

9.2.1 Responsibilities

As a data management platform, OMERO assumes that it is in control of your data in order to help prevent data loss. It assumes that data was copied into the server and only a server administrator or authorized OMERO user would have the rights to do anything destructive to that data.

With in-place import, the data either resides completely outside of OMERO or is shared with other users. This means that the critical, possibly sole, copy of your data must be protected outside of OMERO. This is your responsibility for the lifetime of the data.

9.2.2 Limitations

In-place import is only available on the OMERO server system itself. In other words, using SSH or similar, you will need to shell into the server and run the command-line importer directly. If you are uncomfortable with this, you should let someone else handle in-place importing.

Someone wanting to perform an in-place import **MUST** have:

- a regular OMERO account
- an OS-account with access to bin/omero
- read access to the location of the data
- **write** access to the *ManagedRepository* or one of its subdirectories

The above means that it may be useful to create a single OS account (e.g. “import_user”) which multiple users can log into, and then use their own OMERO accounts to import data. Alternatively, each OMERO user can be given an OS account with access rights to the data storage as well as the managed repository.

At the moment full in-place import functionality is available only under POSIX systems (Unix, Linux, OS X). Under Windows, `ln` and `ln_s` work as expected but `ln_rm` does not automatically remove image files after import.

For soft linking with `--transfer=ln_s` it has been noticed that some plate imports run rather more slowly than usual. Other operations may also be affected. In determining if or how to use in-place import at your high-content screening facility, we thus recommend time profiling with representative data, and alerting us to any significant disappointments.

Also, there is still some data duplication when pyramids are generated. We are hoping to find a work-around for this in the future.
9.2.3 Safety tips

Whether you chose to use the hard- or soft-linking option below, you should take steps to secure files which are in-place imported to OMERO. The best option is making them read-only for both the OMERO user and also for the owner of the data. This means the server cannot accidentally modify the files (e.g. if a client mixes up the file IDs and tries to write to the wrong location) and that the files cannot be removed from the system while OMERO is still using them. Files may not be renamed or otherwise altered such that the OMERO server user cannot find them at the expected location.

If possible, all the files should be added to your regular backup process. If the files for imported images are later removed or corrupted, the result will probably be that while the images remain in their projects or screens with their annotations and basic metadata, they simply cannot be successfully viewed. However, this behavior is not guaranteed, so do not assume that the resulting problems will not extend further. Once the problem is noticed, replacing the original image files from backups, in the same place with the same name, is likely but not guaranteed to fully restore the images and their associated data in OMERO.

9.2.4 Additional setup requirements

In-place import requires additional user and group setup. As no-one should be allowed to log into the account used to install the server, to permit in-place imports you need to create a different user account, allowing someone to log into the server but not accidentally delete any files. Therefore, you should set up an ‘in-place’ user and an ‘in-place’ group and configure a subset of directories under ManagedRepository to let members of that group write to them. The example below details how this was done for one of our test servers in Dundee:

### STATUS BEFORE

```bash
[sysadmin@ome-server omero_system_user]$ umask 0002
[sysadmin@ome-server omero_system_user]$ ls -ltrd ManagedRepository/
drwxrwxr-x 8 omero_system_user omero_system_user 4096 Apr 24 10:13 ManagedRepository/
[sysadmin@ome-server omero_system_user]$ grep inplace /etc/passwd /etc/group
/etc/passwd:inplace_user:x:501:501::/home/inplace_user:/bin/bash
/etc/group:omero_system_user:x:500:inplace_user
[sysadmin@ome-server omero_system_user]$ grep omero_system_user /etc/passwd /etc/group
/etc/passwd:omero_system_user:x:500:500::/home/omero_system_user:/bin/bash
/etc/group:omero_system_user:x:500:inplace_user
[sysadmin@ome-server omero_system_user]$ sudo -u inplace_user -i
[inplace_user@ome-server ~]$ umask 0002
```

### SCRIPT

```
chgrp inplace_user /repositories/binary-repository/ManagedRepository
chmod g+rws /repositories/binary-repository/ManagedRepository
chgrp inplace_user /repositories/binary-repository/ManagedRepository/*
chmod g+rws /repositories/binary-repository/ManagedRepository/*/
chmod g+rws /repositories/binary-repository/ManagedRepository//**
chgrp inplace_user /repositories/binary-repository/ManagedRepository/*/
chgrp inplace_user /repositories/binary-repository/ManagedRepository//**
chgrp inplace_user /repositories/binary-repository/ManagedRepository/****
```

# With the above, newly created directories should be in the inplace group
# As long as the file is readable by omero_system_user, then it should work fine!

### AFTER SCRIPT
### TEST

```
# with default umask this likely has to do
[omero_system_user@ome-server ManagedRepository]$ mkdir inplace.test
[omero_system_user@ome-server ManagedRepository]$ rmdir inplace.test/
```

9.2.5 Getting started

From 5.0.0, the command-line import client has a new help menu which explains the available options:

```
$ bin/omero import --advanced-help
```

ADVANCED OPTIONS:

These options are not intended for general use. Make sure you have read the documentation regarding them. They may change in future releases.

**In-place imports:**

```
--transfer=ARG File transfer method
```

**General options:**

```
upload # Default
upload_rm # Caution! File upload followed by source deletion.
some.class.Name # Use a class on the CLASSPATH.
```

**Server-side options:**

```
ln # Use hard-link.
ln_s # Use soft-link.
ln_rm # Caution! Hard-link followed by source deletion.
cp # Use local copy command.
cp_rm # Caution! Copy followed by source deletion.
```

e.g. $ bin/omero import -- --transfer=ln_s foo.tiff

```
$ ./importer-cli --transfer=ln bar.tiff
$ CLASSPATH=mycode.jar ./importer-cli --transfer=com.example.MyTransfer baz.tiff
```

**Background imports:**

```
--auto_close Close completed imports immediately.
--minutes_wait=ARG Choose how long the importer will wait on server-side processing.
ARG > 0 implies the number of minutes to wait.
ARG = 0 exits immediately. Use a *_completed option to clean up.
ARG < 0 waits indefinitely. This is the default.
--close_completed Close completed imports.
```
--wait_completed Wait for all background imports to complete.

e.g. $ bin/omero import -- --minutes_wait=0 file1.tiff file2.tiff file3.tiff
     $ ./importer-cli --minutes_wait=0 some_directory/
     $ ./importer-cli --wait_completed # Waits on all 3 imports.

File exclusion:
-----------------
--exclude=filename Exclude files based on filename.
--exclude=clientpath Exclude files based on the original path.

e.g. $ bin/omero import -- --exclude=filename foo.tiff # First-time imports
     $ bin/omero import -- --exclude=filename foo.tiff # Second-time skips

Import speed:
--------------
--checksum-algorithm=ARG Choose a possibly faster algorithm for detecting file corruption,
e.g. Adler-32 (fast), CRC-32 (fast), File-Size-64 (fast),
     MD5-128, Murmur3-32, Murmur3-128,
     SHA1-160 (slow, default)

e.g. $ bin/omero import -- --checksum-algorithm=CRC-32 foo.tiff
     $ ./importer-cli --checksum-algorithm=Murmur3-128 bar.tiff

--no-stats-info Disable calculation of minima and maxima when as part of the Bio-Formats reader metadata

e.g. $ bin/omero import -- --no-stats-info foo.tiff
     $ ./importer-cli --no-stats-info bar.tiff

--no-thumbnails Do not perform thumbnailing after import

e.g. $ bin/omero import -- --no-thumbnails foo.tiff
     $ ./importer-cli --no-thumbnails bar.tiff

--no-upgrade-check Disable upgrade check for each import

e.g. $ bin/omero import -- --no-upgrade-check foo.tiff
     $ ./importer-cli --no-upgrade-check bar.tiff

Feedback:
---------
--qa-baseurl=ARG Specify the base URL for reporting feedback

e.g. $ bin/omero import broken_image.tif -- --email EMAIL --report --upload --logs --qa-baseurl=https://qa.staging.openmicroscopy.org/qa
     $ ./importer-cli broken_image.tif --email EMAIL --report --upload --logs --qa-baseurl=https://qa.staging.openmicroscopy.org/qa

Report bugs to <ome-users@lists.openmicroscopy.org.uk>

In versions prior to 5.0.3 this help option is hidden and it can only be accessed using:

$ bin/omero import -- --advanced-help

The option for performing an in-place transfer is --transfer. A new extension point, file transfers allow a choice of which mechanism is used to get a file into OMERO.
$ bin/omero import -- --transfer=ln_s my_file.dv

Using session bba923bb-cf0c-4cf0-80c5-a309be523ad8 (root@localhost:4064). Idle timeout: 10.0 min. Current group: system

...[ main] INFO ome.formats.importer.ImportConfig - OMERO Version: 5.0.0-rc1-DEV-ice35
...[ main] INFO ome.formats.importer.ImportConfig - Bioformats version: 5.0.0-rc1-DEV-ice35
...[ main] INFO formats.importer.cli.CommandLineImporter - Setting transfer to ln_s
...[ main] INFO ome.formats.importer.ImportCandidates - Depth: 4 Metadata Level: MINIMUM
...[ main] INFO ome.formats.importer.ImportCandidates - 1 file(s) parsed into 1 group(s) with 1 call(s) to setId in 98ms. (100ms total) [0 unknowns]
...[ main] INFO ome.formats.OMEROMetadataStoreClient - Attempting initial SSL connection to localhost:4064
...[ main] INFO ome.formats.OMEROMetadataStoreClient - Insecure connection requested, falling back
...[ main] INFO ome.formats.OMEROMetadataStoreClient - Server: 5.0.0
...[ main] INFO ome.formats.OMEROMetadataStoreClient - Client: 5.0.0-rc1-DEV-ice35
...[ main] INFO ome.formats.OMEROMetadataStoreClient - Java Version: 1.7.0_51
...[ main] INFO ome.formats.OMEROMetadataStoreClient - OS Name: Linux
...[ main] INFO ome.formats.OMEROMetadataStoreClient - OS Arch: amd64
...[ main] INFO ome.formats.OMEROMetadataStoreClient - OS Version: 3.8.0-27-generic
...[ main] INFO ome.formats.importer.cli.LoggingImportMonitor - FILESET_UPLOAD_PREPARATION
...[ main] INFO ome.formats.importer.cli.LoggingImportMonitor - FILESET_UPLOAD_START
...[ main] INFO s.importer.transfers.SymlinkFileTransfer - Transferring /tmp/a.fake...
...[ main] INFO ome.formats.importer.cli.LoggingImportMonitor - FILE_UPLOAD_STARTED: /tmp/a.fake
...[ main] INFO ome.formats.importer.cli.LoggingImportMonitor - FILE_UPLOAD_COMPLETE: /tmp/a.fake
...[ main] INFO ome.formats.importer.cli.LoggingImportMonitor - FILESET_UPLOAD_END
Imported pixels: 5001
Other imported objects:
Fileset:4102
Image:5001
...[l.Client-0] INFO ome.formats.importer.cli.ErrorHandler - Number of errors: 0

The only visible difference here is the line:

...formats.importer.cli.CommandLineImporter - Setting transfer to ln_s

Rather than uploading via the OMERO API, the command-line importer makes a call to the system ln command.

9.2.6 Transfer options

Previously, OMERO only offered the option of uploading via the API. Files were written in blocks via the RawFileStore interface. With in-place import, several options are provided out of the box as well as the ability to use your own.

“ln_s” - soft-linking

The most flexible option is soft-linking. For each file, it executes ln -s source target on the local file system. This works across file system boundaries and leaves a clear record of what file was imported:

```
$ OMERO/ManagedRepository/root_0/2014-01/24/10-11-14.947$ ls -ltra
```

```
total 8
lrwxrwxrwx 1 omero omero 11 Jan 24 10:11 my-file.dv -> /home/demo/my-file.dv
```

9.2. In-place import
Here you can see in the imported file set, a soft-link which belongs to the omero user, but which points to a file in the /home/demo directory.

Deleting the imported images in OMERO will delete the soft link but not the original file under /home. This could come as a surprise to users, since the deletion will effectively free no space.

**Warning:** The deletion of the original files under /home (or equivalent) will lead to a complete loss of the data since no copy is held in OMERO. Therefore, this method should only be used in conjunction with a properly managed and backed-up data repository. If the files are corrupted or deleted, there is no way to use OMERO to retrieve them.

### "ln" - hard-linking

The safest option is hard-linking, though it cannot be used across file systems. For each file, it executes `ln source target`. Attempting to hard link across file system boundaries will lead to an error:

```java
java.lang.RuntimeException: transfer process returned 1
```

```java
at omero.api.RawFileStoreDeM.close(RawFileStoreDeM.java:466) ~[blitz.jar:na]
at omero.api.RawFileStorePrxHelper.close(RawFileStorePrxHelper.java:1739) ~[blitz.jar:na]
at omero.api.RawFileStorePrxHelper.close(RawFileStorePrxHelper.java:1701) ~[blitz.jar:na]
at omeroformats.importers.AbstractFileTransfer.cleanupUpload(AbstractFileTransfer.java:123)
at omeroformats.importers.AbstractExecFileTransfer.transfer(AbstractExecFileTransfer.java:69)
at omeroformats.importers.ImportLibrary.uploadFile(ImportLibrary.java:410) ~[blitz.jar:na]
at omeroformats.importers.ImportLibrary.importImage(ImportLibrary.java:465) [blitz.jar:na]
at omeroformats.importers.ImportLibrary.importCandidates(ImportLibrary.java:274) [blitz.jar:na]
at omeroformats.importers.CommandLineImporter.start(CommandLineImporter.java:218) [blitz.jar:na]
at omeroformats.importers.CommandLineImporter.main(CommandLineImporter.java:658) [blitz.jar:na]
```

The safeness of this method comes from the fact that OMERO also has a pointer to the data. Deletion of the original file under /home would leave data in OMERO in place. Again, this could cause a surprise as the space would not be properly freed, but at least there cannot be an accidental loss.

**Warning:** The primary concern with this method is modification of files. If the original data is written by a user, unexpected results could follow in OMERO. See the Safety tips section above for ways around this.

If you are unclear about how hard-linking works, please see the Hard link article on Wikipedia.

The semantics of hard-linking have changed recently on Linux systems with the "protected hardlinks" feature, which is enabled by default and is in use on Ubuntu 14.04, CentOS 7 and other contemporary systems. When you create a hard-link to a file, Linux now requires that you are either the owner of the file, or that you have read-write permissions to the file. Other Unix systems, and older Linux systems, allow a hard-link to be made if you have search access to the file (i.e. you have appropriate read and execute permissions on the directory path containing the file), but do not check the file permissions themselves. See the kernel-hardening mailing list post which describes the change in more detail. The implication for in-place import is that the user performing the import must own or have read-write permissions on the data files being imported in-place.

**"ln_rm" - moving**

Finally, the least favored option is ln_rm. It first performs a hard-link like ln, but once the import is complete it attempts to delete the original file. This is currently in testing as an option for DropBox but is unlikely to be of use to general users. Although this option is more limited than the upload_rm option below it will be much faster.

**"upload_rm" - uploading and deleting**

This option, available from OMERO 5.0.3, is not strictly an in-place option but is detailed here for convenience. It first performs a file upload like default import, but once the import is complete it attempts to delete the original files. It deletes the original files if and only if the import is successful.

**“cp” and “cp_rm” variants**

New in 5.0.7, the cp and cp_rm commands provide the same functionality as ln and ln_rm but perform a copy rather than a link operation. The benefit of a copy is that it works over OS filesystem boundaries while still providing the integrity that ln_s cannot. The primary downside of a raw cp compared to ln is that there is data duplication. cp_rm being very similar to ln_rm usually works around this downside, except in the case of a failed import. Then the duplicated data will remain in OMERO and an explicit cleanup step will need to be taken.

**Your own file transfer**

If none of the above options work for you, it is also possible to write your own implementation of the ome.formats.importer.transfers.FileTransfer class, likely subclassing ome.formats.importer.transfers(AbstractFileTransfer or ome.formats.importer.transfers.AbstractExecFileTransfer. If you do so, please let us know how we might improve either the interface or the implementations that we provide.

Once your implementation has been compiled into a jar and placed in the lib/clients directory, you can invoke it using:

```
4http://en.wikipedia.org/wiki/Hard_link
5http://www.openwall.com/lists/kernel-hardening/2012/02/21/20
```
$ bin/omero import -- --transfer=example.package.ClassName ...

9.2.7 Related advanced options

In addition to the --transfer option in 5.0.0, a number of other advanced options have been added which may be useful for either tweaking import performance or dealing with complicated situations. However, like --transfer, these options should be considered experimental and may change in subsequent releases. Comments and suggestions are very welcome.

Checksums

If you think that calculating the checksums for your large files is consuming too much time, you might want to configure the checksum algorithm used. This can be done with the --checksum_algorithm property. Available options are printed with the --advanced-help option and include Adler-32, CRC-32, MD5-128, Murmur3-32, Murmur3-128, and the default SHA1-160.

DropBox

As described in the scenarios “DropBox import (automatic delete)” and “In-place DropBox import (automatic delete), DropBox can be configured to use any of the options described above. The configuration property to modify is omero.fs.importArgs:

$ bin/omero config set -- omero.fs.importArgs "--transfer=upload_rm"

This will move files into OMERO rather than leaving a copy in the DropBox directory.

$ bin/omero config set -- omero.fs.importArgs "--transfer=ln_rm"

This will also move files into OMERO rather than leaving a copy in the DropBox directory. For this to work, the two directories will need to be on the same file system. This option is much faster than upload_rm. Please read "ln_rm" - moving carefully to ensure you fully understand the implications of using this option.

Warning: Use at your own risk!

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

9.3 Advanced import scenarios

Increasingly users of OMERO are needing to go beyond the traditional “upload via a GUI”-style import model to more powerful methods.

There is a set of requirements for getting data into OMERO that is common to many institutions. Some of the requirements may be mutually exclusive.

• Users need to get data off microscopes quickly. This likely includes not waiting for import to complete. Users will often move data immediately, or even save remotely during acquisition.
• Users would like direct access to the binary repository file-system to read original files for analysis.
• Users would like to view and begin working with images as soon as possible after acquisition.

Below we explain which options are available to you, and why there is a trade-off between the above requirements.
9.3.1 Import overview

In OMERO 4 and earlier versions, data was uploaded to the binary repository in an opaque form; starting with OMERO 5, files are uploaded to the repository in their original form using Bio-Formats.

The “OMERO binary repository” (or repo) is the directory belonging to the OMERO user where files are imported:

- The ManagedRepository directory inside of the repo is where files go during import into OMERO 5. Each user receives a top-level directory inside of “ManagedRepository” which fills with timestamped directories as imports accrue.

- Depending on the permissions of this directory, users may or may not be able to see their imported files. Managing the permissions is the responsibility of the system administrator.

In “normal import”, files are copied to the OMERO binary repo via the API and so can work remotely or locally. In “in-place import”, files are “linked” into place.

**Warning:** In-place import is a new, powerful feature - it is critical that you read and understand the documentation before you consider using it.

9.3.2 Traditional import

**Manual import (GUI)**

This is the standard workflow and the one currently used at the University of Dundee. Users dump data to a shared file-system from the acquisition system, and then use the OMERO.insight client from the lab to import.

**Advantages**

- Users can validate that import worked.
- Failed imports can be repeated and/or reported to QA etc.
- Users do not have to wait for import to be scheduled.
- Import destination is known: Project/Dataset etc.

**Disadvantages**

- Imports can be slow due to the data transfer from file-system to OMERO via the client.
- Users must remember to delete data from the shared file-system to avoid data duplication.
- Users cannot access the OMERO binary repo directly and must download original data via clients for local analysis.

**Manual import (CLI)**

Another typical workflow is that rather than using the GUI, users perform the same procedure as under “Manual import” but with the command-line (CLI) importer.

**Advantages**

- With a CLI workflow, it may be easier for users to connect remotely to kick off an import and to leave it running in the background for a long period of time.

**Disadvantages**

All the same disadvantages apply as under “Manual import (GUI)”.
Cronjob import (manual delete)

For importing via cron, users still dump their data to a shared file-system from the acquisition system. They must have permissions to write to “their” directory which is mapped to a user in OMERO.

A cronjob starts a CLI import, possibly at night. The cronjob could be given admin rights in OMERO to perform an “Import As” for a particular user.

Disadvantages

- If a normal import is used, the cronjob would have to manually delete imported files from their original location to avoid duplication.
- Users cannot work with their data in OMERO until some time after acquisition.
- Failed imports are logged within the managed repository but not yet notified. Logs would probably need to be accessed via a sysadmin/cli. The cronjob could capture stdout and stderr and check for failures.

DropBox import (manual delete)

Similar to the cronjob scenario, DropBox importing requires that users drop their data in “their” directory which has special permissions for writing. The DropBox service monitors those directories for modifications and imports the files on a first-come-first-serve basis.

Advantages

- Users should see their data in OMERO quickly.

Disadvantages

- There is a limitation on the rate of new files in monitored locations.
- There is also a limitation on which file systems can be used. A networked file share cannot be monitored by DropBox.
- Users must manually delete imported files from their DropBox directory to avoid duplication.
- Failed imports are logged within the managed repository but not yet notified. Logs would probably need to be accessed via a sysadmin or through the CLI and searched by the user and file name.

DropBox import (automatic delete)

One option that exists from OMERO 5.0.3 is to have files removed from DropBox automatically after a successful import. This is achieved by first performing an “upload” import from the DropBox directory to the ManagedRepository and then by deleting the data from DropBox if and only if the import was successful. For failed imports, files will remain in the DropBox directories until someone manually deletes them.

Advantages

- For all successful imports, files will be automatically removed from the DropBox directories thus reducing duplication.

9.3.3 In-place import

The following sections outline in-place based scenarios to help you judge if the functionality may be useful for you.
Common 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.

Common disadvantages

- Like the DropBox import scenario above, all in-place imports require the user to have access to the user-based directories under the ManagedRepository. See limitations for more details.
- Similarly, all the following scenarios carry the same burden of securing the data externally to OMERO. This is the primary difference between a normal import and an in-place import: backing up OMERO is no longer sufficient to prevent data loss. The original location must also be secured! This means that users must not move or alter data once imported.

In-place manual import (CLI)

The in-place version of a CLI manual import is quite similar to the normal CLI import, with the primary difference being that the data is not transferred from the shared file-system where the data is initially stored after acquisition, but instead is just “linked” into place.

Advantages

- Local filesystem in-place import is faster than traditional importing, due to the lack of a data transfer.

Disadvantages

- Requires proper security setup as explained above.

In-place Cronjob import

Assuming all the restrictions are met, the cronjob-based workflow above can carry out an in-place import by adding the in-place transfer flag. The advantages and disadvantages are as above.

In-place DropBox import (manual delete)

Just as with the in-place cronjob import, using in-place import for DropBox is as straight-forward as passing the in-place flag. The common advantages and disadvantages of in-place import apply.

In-place DropBox import (automatic delete)

An option that also exists in the in-place scenario is to have files removed from DropBox automatically after a successful import. This is achieved by first performing a “hardlink in-place import” from the DropBox directory to the ManagedRepository and then by deleting the data from DropBox if and only if the import was successful. For failed imports, files will remain in the DropBox directories until someone manually deletes them.

Advantages

- For all successful imports, files will be automatically removed from the DropBox directories.
Disadvantages

- This option is only available if the filesystem which DropBox watches is the same as the file system which the ManagedRepository lives on. This prevents the use of network file systems and similar remote shares.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.
Part III

Developer Documentation
The following documentation is for developers wishing to write OMERO client code or extend the OMERO server. Instructions on downloading\textsuperscript{6}, installation and administering OMERO can be found under the System Administrator Documentation of the main site.

OMERO is an open source client/server system written in Java for visualizing, managing, and annotating microscope images and metadata. The OMERO Application Programming Interface allows clients to be written in Java, Python, C++ or MATLAB. OMERO releases include a Java client OMERO.insight, a Python-based web client OMERO.web and the Command Line Interface as an OMERO development tool, which also uses Python. There is also an ImageJ plugin. OMERO can be extended by modifying these clients or by writing your own in any of the supported languages (see figure). OMERO also supports a Scripting Service which allows Python scripts to be run on the server and called from any of the other clients.

OMERO is designed, developed and released by the Open Microscopy Environment\textsuperscript{7}, with contributions from Glencoe Software, Inc.\textsuperscript{8} OMERO is released under the GNU General Public License (GPL)\textsuperscript{9} with commercial licenses and customization available from Glencoe Software, Inc.\textsuperscript{10}. You can read about how OMERO has developed since the project started in the OMERO version history.

For help with any aspect of OMERO, see details of our forums and mailing lists\textsuperscript{11}.

\textsuperscript{6}http://downloads.openmicroscopy.org/latest/omero/
\textsuperscript{7}http://www.openmicroscopy.org/site
\textsuperscript{8}http://www.glencoesoftware.com/
\textsuperscript{9}http://www.gnu.org/copyleft/gpl.html
\textsuperscript{10}http://www.glencoesoftware.com/
\textsuperscript{11}http://www.openmicroscopy.org/site/community/
INTRODUCTION TO OMERO

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

10.1 What’s new for OMERO 5.2 for developers

- The Version requirements section provides extensive details about which operating systems and dependency versions we intend to support for the life of 5.2 and the likely changes to these for the next major release (currently planned to be 5.3). Most notably for development purposes, we have dropped support for Java 1.6.

10.1.1 Breaking changes

OMERO.web

OMERO.web has undergone a major upgrade, updating jsTree to version 3.0.8 and using json for all tree loading to substantially improve performance.

If you have web code that directly works with the jsTree, you will need to update it according to the jsTree docs\(^1\).

A number of new URLs are available in webclient, with the /api/ prefix that provide json data for the jsTree. However, these may soon be moved to a different Django app and should not yet be considered stable.

Graphs

The API’s request operations Chmod\(^2\), Chgrp\(^3\), Chown\(^4\), Delete\(^5\), and their superclass GraphModify\(^6\), were deprecated in 5.1 and will be removed in 5.3. They are replaced by Chmod2\(^7\), Chgrp2\(^8\), Chown2\(^9\), Delete2\(^10\), and their superclass GraphModify2\(^11\). OMERO developers who have not migrated yet should refer to Using the new 5.1 graph requests and take action before the deprecated features are removed.

Java Gateway

The Java gateway has undergone a major overhaul which should make the development of Java applications much easier. As part of this, pojos has been renamed as omero.gateway.model. We do not intend to modify the methods already available but this

---

1\(^{https://www.jstree.com/}\
2\(^{http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Chmod.html}\
3\(^{http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Chgrp.html}\
4\(^{http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Chown.html}\
5\(^{http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Delete.html}\
6\(^{http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/GraphModify.html}\
7\(^{http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Chmod2.html}\
8\(^{http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Chgrp2.html}\
9\(^{http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Chown2.html}\
10\(^{http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Delete2.html}\
11\(^{http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/GraphModify2.html}\

work should still be considered as under development as we move towards the gateway becoming a stable public API. Feedback as always is welcome. See OMERO Java language bindings for more information or follow the example in minimal-omero-client.

OMERO model

The Rect class has been renamed to Rectangle to match the OME-XML schema. This requires the corresponding change to be made in client software that uses the OMERO.blitz server API to work with ROIs.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

10.2 Installing OMERO from source

10.2.1 Using the source code

The source code of each release of OMERO is available for download from the Source code section of the OMERO download page.

Note: At the moment, this source code bundle does not contain the version of Bio-Formats. To include this version information, you will need to manually copy the ant/gitversion.xml file included in the source code bundle of Bio-Formats for the same release under components/bioformats/ant.

10.2.2 Using the Git source repository

To use the Git source repository, you will need to install Git on your system. See the Using Git section of the Contributing documentation for more information on how to install and configure Git.

The main repository for OMERO is available from https://github.com/openmicroscopy/openmicroscopy. Most OME development is currently happening on GitHub, therefore it is highly suggested that you become familiar with how it works, if not create an account for yourself.

Start by cloning the official repository:

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

Since the openmicroscopy repository now makes use of submodules, you first need to initialize all the submodules:

```
cd openmicroscopy
git submodule update --init
```

Alternatively, with version 1.6.5 of git and later, you can pass the --recursive option to git clone and initialize all submodules:

```
git clone --recursive https://github.com/openmicroscopy/openmicroscopy.git
```

See also:

Using Git Section of the contributing documentation explaining how to use Git for contributing to the source code.

---

12 https://github.com/ome/minimal-omero-client
13 http://downloads.openmicroscopy.org/latest/omero/#code
14 http://www.openmicroscopy.org/site/support/contributing/using-git.html
10.2.3 Building OMERO

To install the dependencies required to run the OMERO.server on Linux or Mac OS X, take a look at the OMERO.server installation page where you will also find links to walk-throughs for specific platforms.

Some environment variables may need to be set up before building the server:

- If the system slice files cannot be found you must set SLICEPATH to point to the slice directory of the Ice installation.

Once all the dependencies and environment variables are set up, you can build the server using:

```python
python build.py
```

or the clients using:

```python
python build.py release-clients
```

See also:

Build System Section of the developer documentation detailing the build system

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

10.3 Build System

Overview

The page goes into details about how the build system is configured.

OMERO mostly uses an Ant-based build with dependency management provided by Ivy. C++ code is built using Cmake and Python uses the traditional distutils/setuptools tools.

10.3.1 Structure of the build

This is an (abbreviated) snapshot of the structure of the filesystem for OMERO:

```bash
OMERO_SOURCE_PREFIX
|-- build.xml ...................... Top-level build file
|-- build.py ...................... Python wrapper to handle OS-specific configuration
|-- omero.class ................... Self-contained Ant launcher
|-- etc ............................ Configuration folder
    |-- grid ....................... Deployment files folder
    |   |-- ivysettings.xml ........ Main Ivy configuration file
    |   |-- hibernate.properties
    |   |-- build.properties
    |   |-- logback.xml
    |   |-- omero.properties
    \-- profiles
```

---

16 http://ant.apache.org
17 http://ant.apache.org/ivy
|-- examples ....................... User examples
|\components
||--<component-name> .............. Each component has this same basic structure.
| ||-- build.xml ................... Build file
| ||-- ivy.xml ..................... Jar dependencies
| ||-- test.xml .................... Test dependencies
| ||-- src ........................ Source code
| ||-- test ........................ Test source code and test resources
| |\-- target ...................... Build output (deleted on clean)
| |
\NOTABLE COMPONENTS

|-- model ............................ The model component is special in that it produces
| a jar specific to your database choice: model-psql.jar
| The generated 'ome.model.*' files contain Hibernate
| annotations for object-relational mapping.

|-- blitz ............................ The blitz component also performs code generation
| producing artifacts for Java, Python, and C++.
| and other build tools.

|--tools ............................. Other server-components with special build needs.
| |--build.xml .................... Build scripts
| |
| \--<tool-name>
| ||--build.xml ................... Build file
| |\--ivy.xml ..................... Jar dependencies
| |
| \--antlib ............................ Special component which is not built, but referenced by the build
| |
| \--resources ........................ Build resources
| |--global.xml ................... Global build properties
| |--hibernate.xml
| |--lifecycle.xml .............. Ivy-related targets
| \--version.xml ................ Version properties

\Note: User examples are explained under Working with OMERO

Unfortunately, just the above snapshot of the code repository omits some of the most important code. Many megabytes of source code is generated both by our own DSLTask\(^\text{18}\) as well as by the ice\(^\text{19}\) slice2java, slice2cpp, and slice2py code generators. These take an intermediate representation of the OME-Model\(^\text{20}\) and generate our OME-Remote Objects. This code is not available in git, but once built, can be found in all the directories named “generated”.

\subsection{10.3.2 Build tools}

\section{Ant}

./build.py is a complete replacement for your local ant install. In many cases, you will be fine running ant. If you have any issues (for example OutOfMemory), please use ./build.py instead. However, only use one or the other; do not mix calls between the two.

The main build targets are defined in the top-level build.xml file. All available targets can be listed using:

\footnotesize{\begin{verbatim}
18https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/dsl
19https://zeroc.com
20http://www.openmicroscopy.org/site/support/ome-model/ome-xml/
Each of the component contains a build.xml and can be built directly using:

./build.py -f components/server/build.xml

This will call the default dist target for each component.

Ivy

The build system uses Ivy\textsuperscript{21} 2.3.0 as the dependency manager. The general Ivy configuration is defined in a settings file\textsuperscript{22} located under etc/ivysettings.xml\textsuperscript{23}.

In order to determine the transitive closure of all dependencies, Ivy resolves each ivy.xml file and stores the resolved artifacts in a cache\textsuperscript{24} to speed up other processes. The OMERO build system defines and uses two kinds of caches:

1. the local dependencies cache under lib/cache is used by most resolvers
2. Maven resolvers use the Maven cache under ~/.m2/repository

Note: When the Ivy configuration file or the version number is changed, the cache can become stale. Calling ./build.py clean from the top-level build will delete the content of the local cache.

Resolvers\textsuperscript{25} are key to how Ivy functions. Multiple dependency resolvers can be defined fine-grained enough to resolve an individual jar in order to pick up the latest version of any library from a repository\textsuperscript{26}, a generic URL\textsuperscript{27} or from the local filesystem\textsuperscript{28}.

Since OMERO 5.1.3, the remote repository resolvers are set up to resolve transitive dependencies.

The OMERO build system uses by default a chain resolver\textsuperscript{29} called omero-resolver which resolves the following locations in order:

1. target/repository which contains most artifacts published by the build system in the install step of the lifecycle
2. the local dependency repository under lib/repository
3. the local Maven cache under ~/.m2/repository
4. the Maven central repository\textsuperscript{30}
5. the OME artifactory\textsuperscript{31}

Bio-Formats dependencies are resolved using a specific chain resolver\textsuperscript{32} called ome-resolver which resolves the following locations in order:

1. the local Maven cache under ~/.m2/repository
2. the OME artifactory\textsuperscript{33}

To define its dependencies, each component uses a top-level Ivy file\textsuperscript{34}, ivy.xml, for the build and optionally another Ivy file, test.xml, for the tests.

The OMERO build system defines and uses four types of Ivy configurations\textsuperscript{35}:

\textsuperscript{21}http://ant.apache.org/ivy
\textsuperscript{22}http://ant.apache.org/ivy/history/2.3.0/settings.html
\textsuperscript{23}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/etc/ivysettings.xml
\textsuperscript{24}http://ant.apache.org/ivy/history/2.3.0/settings/caches/cache.html
\textsuperscript{25}http://ant.apache.org/ivy/history/2.3.0/settings/resolvers.html
\textsuperscript{26}http://ant.apache.org/ivy/history/2.3.0/resolver/ibiblio.html
\textsuperscript{27}http://ant.apache.org/ivy/history/2.3.0/resolver/url.html
\textsuperscript{28}http://ant.apache.org/ivy/history/2.3.0/resolver/filesystem.html
\textsuperscript{29}http://ant.apache.org/ivy/history/2.3.0/resolver/chain.html
\textsuperscript{30}http://central.sonatype.org
\textsuperscript{31}http://artifacts.openmicroscopy.org
\textsuperscript{32}http://ant.apache.org/ivy/history/2.3.0/resolver/chain.html
\textsuperscript{33}http://artifacts.openmicroscopy.org
\textsuperscript{34}http://ant.apache.org/ivy/history/2.3.0/ivyfile.html
\textsuperscript{35}http://ant.apache.org/ivy/history/2.3.0/ivyfile/configurations.html
1. **build**: defines dependencies to be used for building
2. **server**: defines dependencies to be bundled under `lib/server`
3. **client**: defines dependencies to be bundled under `lib/client`
4. **test**: defines dependencies to be used for running the tests

While building, most Java components follow the same lifecycle defined in `lifecycle.xml`. The default `dist` target for each component calls each of the following steps in order:

1. **retrieve**: retrieve\(^{37}\) the resolved dependencies and copy them under `target/libs`
2. **prepare**: prepare various resources (property files, `lib/logback-build.xml`\(^{38}\))
3. **generate**: copy all resources from the previous step for compilation
4. **compile**: compile the source files into the destination repository
5. **package-extra**: package the sources and the Javadoc into Jar files for publication
6. **package**: package the compiled classes into a Jar file for publication
7. **install**: convert the component Ivy file into a pom file using `makepom`\(^{39}\) and `publish`\(^{40}\) the component artifacts

Individual components can override the content of this default lifecycle via their `build.xml`.

### OmeroTools

The Ant\(^{41}\) build alone is not enough to describe all the products which get built. Namely, the builds for the non-Java components stored under `components/tools`\(^{42}\) are a bit more complex. Each tools component installs its artifacts to the tools/target directory which is copied on top of the dist top-level distribution directory.

### Jenkins

The OME project currently uses Jenkins\(^{43}\) as a continuous integration server available here\(^{44}\), so many binary packages can be downloaded without compiling them yourself. See the Continuous Integration documentation\(^{45}\) for further details.

### 10.3.3 Server build

The default ant target (`build-default`) will build the OMERO system and copy the necessary components for a binary distribution to the `dist` directory. Below is a comparison of what is taken from the build, where it is put, and what role it plays in the distribution.

<table>
<thead>
<tr>
<th><strong>OMERO_SOURCE_PREFIX</strong></th>
<th><strong>OMERO_SOURCE_PREFIX/dist</strong></th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>components/blitz/target/blitz.jar</td>
<td>lib/server</td>
<td>Primary Ice servants</td>
</tr>
<tr>
<td>components/blitz/target/server.jar</td>
<td>lib/server</td>
<td>Primary server logic</td>
</tr>
<tr>
<td>components/tools/OmeroCpp/lib*</td>
<td>lib/python</td>
<td>Native shared libraries</td>
</tr>
<tr>
<td>components/tools/OmeroPy/build/lib</td>
<td>lib/client &amp; lib/server</td>
<td>Python libraries</td>
</tr>
<tr>
<td>lib/repository/&lt;some&gt;</td>
<td>etc/</td>
<td>Libraries needed for the build</td>
</tr>
<tr>
<td>etc/</td>
<td>etc/</td>
<td>Configuration</td>
</tr>
<tr>
<td>sql/*.sql</td>
<td>sql/</td>
<td>SQL scripts to prepare the database</td>
</tr>
</tbody>
</table>

Note: By default, **OMERO C++ language bindings** are not built. Use `build-all` for that.

These files are then zipped to `OMERO.server-<version>.zip` via `release-zip`.  

---

\(^{36}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/antlib/resources/lifecycle.xml  
\(^{37}\)http://ant.apache.org/ivy/history/2.3.0/use/retrieve.html  
\(^{38}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/lib/logback-build.xml  
\(^{39}\)http://ant.apache.org/ivy/history/2.3.0/use/makepom.html  
\(^{40}\)http://ant.apache.org/ivy/history/2.3.0/use/publish.html  
\(^{41}\)http://ant.apache.org  
\(^{42}\)https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/tools  
\(^{43}\)http://jenkins-ci.org  
\(^{44}\)https://ci.openmicroscopy.org/  
\(^{45}\)http://www.openmicroscopy.org/site/support/contributing/ci-omero.html
10.3.4 Coupled development

Since OMERO 5.1.3, Bio-Formats is decoupled from the OMERO build system which consumes Bio-Formats artifacts from the OME Maven repository via Ivy\(^{46}\).

While this decoupling matches most of the development use cases, it is sometimes necessary to work on coupled Bio-Formats and OMERO branches especially during breaking changes of the OME Data Model or the Bio-Formats API.

The general rule for coupled branches is to build each component in their dependency order and use the local Maven repository under ~/.m2/repository to share artifacts.

Building Bio-Formats

From the top-level folder of the Bio-Formats repository,

1. adjust the version of Bio-Formats which will be built, installed locally and consumed by OMERO e.g. for 5.2.0-SNAPSHOT:

   $ ./tools/bump_maven_version.py 5.2.0-SNAPSHOT

2. run the Maven command allowing to build and install the artifacts under the local Maven cache:

   $ mvn clean install

Building OMERO

From the top-level folder of the OMERO repository,

1. adjust the versions.bioformats property under etc/omero.properties\(^{47}\) to the version chosen for the Bio-Formats build

2. adjust the ome.resolver property under etc/build.properties\(^{48}\) to be ome-resolver

3. run the build system as usual:

   $ ./build.py build-dev

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

10.4 Working with OMERO

This page describes various tools and resources useful for working with the OMERO API, as well as some tips on setting up your working environment. It should be useful to client developers working in any of the supported languages. For language specific info, see the following links: OMERO Java language bindings, OMERO Python language bindings, OMERO C++ language bindings, OMERO MATLAB language bindings.

10.4.1 OMERO.clients

The OMERO model is implemented as a relational PostgreSQL database on the OMERO.server and mapped to code-generated model objects used by the clients in the various supported languages (linked above). The OMERO API consists of a number of

\(^{46}\)http://ant.apache.org/ivy
\(^{47}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/etc/omero.properties
\(^{48}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/etc/build.properties
services for working with these objects and associated binary data. Typically, clients will use various stateless services to query the OMERO model and then use the stateful services for exchange of binary data or image rendering.

A typical client interaction might have an outline such as:

- Log in to OMERO, obtaining connection and ‘service factory’.
- Use the stateless ‘Query Service’ or ‘Container Service’ to traverse Projects, Datasets and Images.
- Use the stateful ‘Rendering Engine’ or ‘Thumbnail Service’ to view images.
- Use the stateful ‘Raw Pixels Service’ or ‘Raw File Store’ to retrieve pixel or file data for analysis.
- Create new Annotations or other objects and save them with the stateless ‘Update Service’.
- Close stateful services to free resources and close the connection.

OMERO clients use a common ‘gateway’ to communicate with an OMERO server installation and allow the user to import, display, edit, and manage server data. The OMERO team has developed a suite of clients (see OMERO clients overview), but the open source nature of the OMERO project also allows developers to create their own, customized clients. If you are interested in doing this, further information is available on Developing OMERO clients.

10.4.2 OMERO server

Although most interactions with OMERO can be achieved remotely, you will generally find it easier to have the server installed on your development machine, particularly if you are going to be doing a lot of OMERO development. This gives you local access to the database, binary repository, logs etc. and means you can work ‘off-line’.

Even if the server you are connecting to is remote, you will still want to have the server package available locally, so as to give you the command line tools, Python libraries, etc. It is important that all OMERO server and client libraries you use are the same OMERO version.

You may wish to work with the most recent OMERO release, or alternatively you can use the latest development code. Instructions on how to download or check out the code can be found on the main downloads page.

Regular builds of the server are performed by Jenkins including generated Javadocs. See the contributing developer continuous integration documentation for more information.

10.4.3 Environment variables

In addition to the install instructions, you might find it useful to set the following variables:

- OMERO_HOME: should be set to the directory containing the OMERO distribution or build (note that we do not recommend setting this variable for production servers, it is for development usage only - see Setting the OMERO_HOME environment variable)

```
# E.g. If you built the server yourself
export OMERO_PREFIX=~/Desktop/OMERO/dist
# Or you downloaded a release package
export OMERO_PREFIX=~/Desktop/OMERO.server-5.0.x
```

- Add the /bin/ directory to your PATH - allows you to call the ‘omero’ command from anywhere

```
export PATH=$PATH:OMERO_PREFIX/bin/
```

- For Python developers, set your PYTHONPATH as follows

```
export PYTHONPATH=$PYTHONPATH:OMERO_PREFIX/lib/python/
```

---

49 http://downloads.openmicroscopy.org/latest/omero/
50 http://jenkins-ci.org
51 http://www.openmicroscopy.org/site/support/contributing/ci-omero.html
Now checkout the CLI.

```
$ omero -h
```

### 10.4.4 Network hopping for laptops

By default OMERO will bind to all available interfaces. On a laptop this has the undesirable effect of requiring an OMERO restart when changing network connections, e.g. from a home to a work network connection. To avoid this, it is possible to bind only on the localhost interface which will never change IP address.

```
$ omero config set Ice.Default.Host 127.0.0.1
# Restart to activate the new setting
$ omero admin restart
```

**Note:** Be warned, if doing this, it will no longer be possible to connect to the OMERO server instance from anywhere except the local machine.

### 10.4.5 Database access

It is useful to be able to directly query or browse the OMERO PostgreSQL database, which can be achieved with a number of tools. E.g.

- **psql** - this command line tool should already be installed. Depending on your permissions, you may need to connect as the 'postgres' user:

  ```
  $ sudo -u postgres psql omero
  Password: # sudo password
  omero=# \d; # give a complete list of tables and views
  omero=# \d annotation; # list all the columns in a particular table
  omero=# select id, discriminator, ns, textValue, file from annotation order by id desc; # query
  ```

- **pgAdmin**[^52] is a free, cross platform GUI tool for working with PostgreSQL.

### 10.4.6 OMERO model

You can browse the OMERO model in a number of ways, one of which is by looking at the database itself (see above). Another is via the OMERO model API[^53] documentation.

However, due to the complexity of the OMERO model, it is helpful to have some starting points (follow links below to the docs themselves).

**Note:** These figures show highly simplified outlines of various model objects.

### Projects, datasets and images


[^57]: [http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/model/ProjectDatasetLink.html](http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/model/ProjectDatasetLink.html)
Projects, Datasets, Images and a number of other entities can be linked to Annotations (abstract superclass) via specific links (ProjectAnnotationLink, DatasetAnnotationLink etc). Annotation subclasses such as CommentAnnotation, FileAnnotation etc. are stored in a single database table in OMERO (all Annotations have unique ID).

Images

Images in OMERO are made up of many entities. These include core image components such as Pixels and Channels, as well as a large number of additional metadata objects such as Instrument (microscope), Objective, Filters, Light Sources, and Detectors.

10.4.7 Working with the OMERO model objects

For detailed information see OME-Remote Objects and Developing OMERO clients pages.

Objects that you wish to work with on the client must be loaded from OMERO, with the query defining the extent of any data graph that is “fetched”.

The OMERO Application Programming Interface supports two principle ways of querying OMERO and retrieving the objects. You can write SQL-like queries using the query service (uses “HQL”) or you can use one of the other services that already has suitable queries. Using the query service is very flexible but it requires detailed knowledge of the OMERO model (see above) and is susceptible to any change in the model.

For example, to load a specific Project and its linked Datasets you could write a query like this:

```python
queryService = session.getQueryService()
params = omero.sys.Parameters()
params.map = {"pid": rlong(projectId)}
query = "select p from Project p left outer join fetch p.datasetLinks as links left outer join fetch links.child as dataset where p.id=:pid"
project = queryService.findByQuery(query, params)
for dataset in project.linkedDatasetList:
    print dataset.getName().getValue()
```

Or use the Container Service like this:

---

59http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/model/Annotation.html  
60http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/model/ProjectAnnotationLink.html  
61http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/model/DatasetAnnotationLink.html  
63http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/model/FileAnnotation.html  
64http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/model/Pixels.html  
65http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/model/Channel.html
containerService = session.getContainerService()
project = containerService.loadContainerHierarchy("Project", [projectId], True)
for dataset in project.linkedDatasetList:
    print dataset.getName().getValue()

For a list of the available services, see the OMERO Application Programming Interface page.

### 10.4.8 Examples

**HQL examples**

HQL is used for Query Service queries (see above). Some examples, coupled with the references for the OMERO model and HQL syntax\(^\text{66}\) should get you going, along with notes about object loading on the OME-Remote Objects page.

**Note:** If possible, it is advisable to use an existing API method from one of the other services (as for the container service above).

Although it is possible to place query parameters directly into the string, it is preferable (particularly for type-checking) to use the omero.sys.Parameters object:

queryService.findByQuery("from PixelsType as p where p.value='\$s'" % pType, None)

psql queries

Below are a number of example psql database queries:

# list any images that do not have pixels:
```sql
select id, name from Image i where i.id not in (select image from Pixels where image is not null) order by i.id desc;
```

```sql
select id, name, ome_perms(permissions) from experimentergroup;
```

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>ome_perms</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>system</td>
<td>-rw----</td>
</tr>
<tr>
<td>1</td>
<td>user</td>
<td>-rwr-r-</td>
</tr>
<tr>
<td>2</td>
<td>guest</td>
<td>-rw------</td>
</tr>
<tr>
<td>3</td>
<td>JRS-private</td>
<td>-rw------</td>
</tr>
<tr>
<td>4</td>
<td>JRS-read-only</td>
<td>-rwr---</td>
</tr>
</tbody>
</table>

```sql
select id, name, path, owner_id, group_id, ome_perms(permissions) from originalfile order by id desc limit 100;
```

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>path</th>
<th>owner_id</th>
<th>group_id</th>
<th>ome_perms</th>
</tr>
</thead>
<tbody>
<tr>
<td>113</td>
<td>stdout</td>
<td>/Users/will/omero/tmp/omero_will/75270/processuLq8fd.dir/out</td>
<td>207</td>
<td>208</td>
<td>-rw------</td>
</tr>
<tr>
<td>110</td>
<td>imagesFromRois.py</td>
<td>ScriptName061ea79c-f98c-447b-b720-d17003d6a72f</td>
<td>0</td>
<td>0</td>
<td>-rw------</td>
</tr>
</tbody>
</table>

# find all annotations on Image ID=2
```sql
select * from annotation where id in (select child from imageannotationlink where parent = 2) ;
```

# trouble-shooting postgres
```sql
select * from pg_stat_activity ;
```

You can use the omero omero hql command to query a remote OMERO database, entering your login details when requested.
Note: Because you will be querying the database under a particular login, the entries returned will be subject to the permissions of that login.

bin/omero hql -q --limit=10 "select name from OriginalFile where id=4106"
bin/omero hql -q --limit=10 "select id, textValue, file from Annotation a order by a.id desc"
bin/omero hql -q --limit=10 "select id, textValue from TagAnnotation a order by a.id desc"
bin/omero hql -q --limit=100 "select id, owner.id, started, userAgent from Session where closed is null"

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

10.5 Running and writing tests

The following guidelines apply to tests in both the Java and Python test components. However, some of the presented options apply to only one or the other.

The default build target does not compile all the required testing resources. You should run test-compile (or build-dev if you are using Eclipse) first:

```
./build.py build-default test-compile
```

You must rebuild the test-compile target if you subsequently modify any of the Java tests.

Note: The OMERO C++ components and tests are under heavy development, and are not compiled or run by the targets mentioned on this page.

10.5.1 Running tests

Running unit tests

The unit testing framework is fairly simple. Only methods which contain logic written within OMERO are tested. This means that framework functionality like remoting or the Hibernate layer is not tested. This is a part of integration testing (see below).

You can run the unit tests for any component from its directory by entering:

```
./build.py -f components/<component>/build.xml test
```

The same can be done for all components using:

```
./build.py test-unit
```

Note that for tests written in Python the package pytest must be installed, see Writing Python tests. Also note that some Python tests are excluded by default, see Using markers in OmeroPy tests for more details.

Running integration tests

Integration testing is a bit more complex because of the reliance on a database, which is not easily mockable. All Hibernate-related classes are tested in integration mode.

The tests require a fast computer. Running all the integration tests places several restrictions on the environment:

- There must be a running OMERO database.
- An OMERO.server instance must be running.
Integration tests assume that:

- **ICE_CONFIG** has been properly set. The contents of the etc/ice.config file should be enough to configure a running server for integration testing. This means that code creating a client connection as outlined in Developing OMERO clients should execute without errors.

- An OMERO.server instance is running on the host and port specified in the **ICE_CONFIG** file.

If any of the tests fail with a user authentication exception (or omero.client throws an exception), a new ice.config file can be created and pointed to by the **ICE_CONFIG** environment variable. Most likely the first settings that will have to be put there will be omero.user and omero.pass.

**Running all tests**

To run all the integration tests, use

```
./build.py test-integration
```

Note that some Python tests are excluded by default, see Using markers in OmeroPy tests for more details.

**Component tests**

Running an integration test suite for an individual component can be done explicitly via:

```
./build.py -f components/<component>/build.xml integration
```

Results are placed in components/<component>/target/reports.

**Individual tests**

Alternatively, you can run individual tests which you may currently be working on. This can be done using the test target. For example:

```
./build.py -f components/tools/OmeroJava/build.xml test -DTEST=integration/chown/PermissionsTest
./build.py -f components/tools/OmeroPy/build.xml test -DTEST=test/integration/test_admin.py
```

**Running Java tests**

**Individual test class methods**

Individual OmeroJava test class methods (or a comma-separated list of methods) can be run using the `-DMETHODS` parameter together with the test target. The test method must be provided in the fully qualified name form (`-Dpackage.class.method`).

```
./build.py -f components/tools/OmeroJava/build.xml test -DMETHODS=integration.chgrp.AnnotationMoveTest.testMoveTaggedImage
```

**Individual test groups**

To run individual OmeroJava test groups (or comma-separated sets of groups) of tests, the `-DGROUPS` parameter can be used together with the test target

```
./build.py -f components/tools/OmeroJava/build.xml test -DGROUPS=integration.chgrp.
```
Using Eclipse to run tests

To facilitate importing OMERO components into Eclipse, there are .project and .classpath-template files stored in each component directory (e.g. common's .classpath and common's .project).

There are also top-level .classpath and .project files which allow for importing all components as a single project, but this approach requires more memory and does not clearly differentiate the classpaths, and so can lead to confusion.

Before importing any component as a project into Eclipse, a successful build has to have taken place:

./build.py

This is for two reasons. Firstly, the Eclipse projects are not configured to perform the code generation needed. The build.py command creates the directory:

<component>/target

which will be missing from any Eclipse project you open before building the source.

Secondly, Ivy is used to copy all the jar dependencies from OMERO_SOURCE_PREFIX/lib/repository to <component>/target/libs, which is then used in the Eclipse .classpath files.

If Eclipse ever gets out of sync after the first build, ./build.py build-eclipse can be used to quickly synchronize.

A prerequisite of running unit and integration tests in the Eclipse UI is having the TestNG plug-in installed and working (help available on the TestNG site).

Running the unit tests under Eclipse requires no extra settings and is as easy as navigating to the package or class context menu Run As or Debug As, then selecting TestNG.

Integration tests require the ICE_CONFIG environment variable to be available for the Eclipse-controlled JVM. This can be done by editing Debug/Run configurations in Eclipse. After navigating to the Debug (or Run) Configurations window, the Environment tab needs to be selected. After clicking New, ICE_CONFIG can be defined as a path to the ice.config file. This setting needs to be defined per package, class or method.

By using the “debug” target from templates.xml, it is possible to have OMERO listen on port 8787 for a debugging connection.

bin/omero admin stop
bin/omero admin start debug

Then in Eclipse, you can create a new “Debug” configuration by clicking on Remote Java Application, and setting the port to 8787. These values are arbitrary and can be changed locally.

Keep in mind:

- The server will not start up until you have connected with Eclipse. This is due to the “suspend=y” clause in templates.xml.
  If you would like the server to start without you connecting, use “suspend=n”.
- If you take too much time examining your threads, your calls may throw timeout exceptions.

---

67 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/.classpath-template
68 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/.project
69 http://testng.org/doc/eclipse.html
Running Python tests

Using markers in OmeroPy tests

Tests under OmeroPy can be included or excluded according to markers defined in the tests. This can be done by using the `-DMARK` option. For example, to run all the integration tests marked as `broken`:

```
./build.py -f components/tools/OmeroPy/build.xml integration -DMARK=broken
```

By default tests marked as `long_running` and `broken` are excluded and so the following two builds are equivalent:

```
./build.py -f components/tools/OmeroPy/build.xml integration
./build.py -f components/tools/OmeroPy/build.xml integration -DMARK="not (long_running or broken)"
```

In order to run all tests, including `long_running` and `broken`, an empty marker must be used:

```
./build.py -f components/tools/OmeroPy/build.xml integration -DMARK=
```

See also:

Marking OmeroPy tests

Running tests directly

When writing tests it can be more convenient, flexible and powerful to run the tests from `components/tools/OmeroPy` using `setup.py test`. Since Python is interpreted, tests can be written and then run without having to rebuild or restart the server. A few basic options are shown below.

```
-t <test_path>, --test-path <test_path>
```

This option specifies the test suite to run. For instance to run a single test file:

```
cd components/tools/OmeroPy
./setup.py test -t test/integration/test_admin.py
```

Or to run all tests under a given folder:

```
cd components/tools/OmeroPy
./setup.py test -t test/integration/clitest
```

```
-k <string>
```

This option will run all integration tests containing the given string in their names. For example, to run all the tests under `test/integration` with `permissions` in their names:

```
./setup.py test -t test/integration -k permissions
```

This option can also be used to run a named test within a test module:

```
./setup.py test -t test/integration/test_admin.py -k testGetGroup
```

```
-m <marker>
```

This option will run integration tests depending on the markers they are decorated with. Available markers can be listed using the `--markers` option. For example, to run all integration tests excluding those decorated with the marker `long_running`:

```
https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/tools/OmeroPy
```
```bash
./setup.py test -t test/integration -m "not long_running"
```

`--markers`
This option lists available markers for decorating tests:

```bash
./setup.py test --markers
```

`-s`
This option allows the standard output to be shown on the console:

```bash
./setup.py test -t test/integration/test_admin.py -s
```

`-h`, `--help`
This option displays the full list of available options:

```bash
./setup.py test -h
```

To make use of the more advanced options available in `pytest` that are not accessible using `setup.py test`, the `py.test` script can be used directly. To use this `PYTHONPATH` must contain the path to the OMERO Python libraries, see `OMERO Python language bindings` as well as the path to the OMERO Python test library\(^1\). Alternatively, the `pytest` plugin `pytest-pythonpath`\(^2\) can be used to add paths to `PYTHONPATH` specifically for `py.test`.

`--repeat <number>`
This option allows to repeat tests for `number` occurrences:

```bash
py.test --repeat 20 test/unit/fstest
```

`-h`, `--help`
This option displays the full list of options:

```bash
py.test --help
```

and http://pytest.org/latest/usage.html for more help in running tests.

### Failing tests

The `test.with.fail` ant property is set to `false` by default, which prevents test failures from failing the build. However, it can instead be set to `true` to allow test failures to fail the build. For example:

```bash
./build.py -Dtest.with.fail=true integration
```

Some components might provide individual targets for specific tests (e.g. OmeroJava provides the `broken` target for running broken tests). The `build.xml` file is the reference in each component.

## 10.5.2 Writing tests

### Writing Java tests

For more information on writing tests in general see http://testng.org. For a test to be an “integration” test, place it in the “integration” TestNG group. If a test is temporarily broken, add it to the “broken” group:

\(^1\)https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/tests/python

\(^2\)https://pypi.python.org/pypi/pytest-pythonpath
@Test(groups = {"integration", "broken"})
public void testMyStuff() {
}

Tests should be of the **Acceptance Test** form. The ticket number for which a test is being written should be added in the TestNG annotation:

```java
@Test(groups = "ticket:60")
```

This works at either the method level (see [SetsAndLinksTest.java](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/model/test/ome/model/utests/SetsAndLinksTest.java)) or the class level (see [UniqueResultTest.java](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/test/ome/server/itests/query/UniqueResultTest.java)).

The tests under `components/tools/OmeroJava/test` will be the starting point for most Java-client developers coming to OMERO. An example skeleton for an integration test looks similar to

```java
@Integration(groups = "integration")
public class MyTest {
    omero.client client;

    @BeforeClass
    protected void setup() throws Exception {
        client = new omero.client();
        client.createSession();
    }

    @AfterClass
    protected void tearDown() throws Exception {
        client.closeSession();
    }

    @Test
    public void testSimple() throws Exception {
        client.getSession().getAdminService().getEventContext();
    }
}
```

### Writing Python tests

To write and run Python tests you first need to install `pytest`:

```bash
pip install pytest
```

For more information on writing tests in general see [http://pytest.org](http://pytest.org).

Similar to the OmeroJava tests, the tests under `components/tools/OmeroPy/test`, `components/tools/OmeroFS/test` and `components/tools/OmeroWeb/test` will be the starting point for most Python-client developers coming to OMERO. Integration tests should be placed under the integration subfolders. The file names must begin with `test_` for the tests to be found by `pytest`.

---

10.5. Running and writing tests
import omero

class TestExample(object):
    def setup_method(self, method):
        client = new omero.client()
        client.createSession()

    def teardown_method(self, method):
        client.closeSession()

    def testSimple():
        ec = client.getSession().getAdminService().getEventContext()
        assert ec, "No EventContext!"

Marking OmeroPy tests

Methods, classes and functions can be decorated with pytest markers to allow for the selection of tests. pytest provides some predefined markers and functions can be simply defined as they are used. However, to centralize the use of custom markers they should be defined in components/tools/pytest.ini.

To view all available markers the --markers option can be used with setup.py test or py.test as detailed in Running tests directly.

There are a small number of custom markers defined:

**long_running** Used to mark tests as long running. These are tests that typically take 10 minutes or more and are excluded from the daily integration builds.

**broken** Used to mark broken tests. These are tests that fail consistently with no obvious quick fix. Broken tests are excluded from the main integration builds and instead are run in a separate daily build. broken markers should have a reason, an associated Trac ticket number or both. If there are multiple associated tickets then a comma-separated list should be used.

**intermittent** Used to mark tests that fail intermittently. intermittent markers should have a reason, an associated Trac ticket number or both. If there are multiple associated tickets then a comma-separated list should be used. Tests marked as intermittent are included in the daily integration builds.

import pytest
class TestExample2(object):
    @pytest.mark.broken(reason="Asserting false", ticket="12345,67890")
    def testBroken():
        assert False, "Bound to fail"

    @pytest.mark.long_running
    def testLongRunning():
        self.somePotentiallyLongCall()

    @pytest.mark.intermittent(reason="Occasionally times out", ticket="98765")
    def testIntermittent():
        self.somePotentiallyIntermittentCall()

Using the Python test library

The OMERO Python test library defines an abstract ITest class that implements the connection set up as well as many methods shared amongst all Python integration tests.

---

79 https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/tools/pytest.ini
80 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tests/python/library/__init__.py
Each concrete instance of the ITest will initiate a connection to the server specified by the ICE_CONFIG environment variable at the setup_class() level. The following objects are created by ITest.setup_class() and shared by all test methods of this class:

- self.root is a client for the root user
- self.group is a new group which permissions are set to ITest.DEFAULT_PERMS by default. Overriding DEFAULTS_PERMS in a subclass of ITest means the group will be created with the new permissions.
- self.user is a new user and member of self.group
- self.client is a client for the self.user created at class setup.

Additionally, for the self.client object, different shortcuts are available:

- self.sf is the non-root client session
- self.update is the update service for the non-root client session
- self.query is the query service for the non-root client session
- self.ctx is the event context for the non-root client session. Note this corresponds to the context at creation time and should be refreshed if the context is modified.

The example below inherits the ITest class and would create a read-write group by default

```python
from library import ITest
class TestExample(ITest):
    DEFAULT_PERMS = 'rwrw-'  # Override default permissions
    def test1():
        doAction(self.sf)
```

New user and groups can be instantiated by individual tests using the ITest.new_user() and ITest.new_group() methods:

```python
def testNewGroupOwner():
    new_group = self.new_group(perms='rwa--')
    new_owner = self.new_user(group=new_group, owner=True)
    assert new_owner.id.val, "No EventContext!"
```

New clients can be instantiated by individual tests using the ITest.new_client() or ITest.new_client_and_user() methods:

```python
def testNewClient():
    new_client = self.new_user_and_client()
    ec = new_client.getSession().getAdminService().getEventContext()
    assert ec, "No EventContext!"
```

Images can be imported using the ITest.importSingleImage() or ITest.importSingleImageWithCompanion() or ITest.importMIF() methods:

```python
def testImportMIF():
    images = self.importMIF(2)
    assert len(images) == 2
```
Writing OMERO.web tests

For OMERO.web integration tests, the OMERO.web test library\(^1\) defines an abstract IWebTest class that inherits from ITest and also implements Django clients at the class setup using the Django testing tools\(^2\).

On top of the elements created by ITest\(\cdot\)setup\_class\(), the IWebTest class creates:

- self.django\_root\_client is a Django test client for the root user
- self.django\_client is a client for the new user created at the class setup.

```python
from weblibrary import IWebTest
class TestExample(IWebTest):
    def testSimple():
        self.django_client.post('/login/', {'username': 'john'})
```

New Django test clients can be instantiated by individual tests using the IWebTest\(\cdot\)new\_django\_client\() method:

```python
def testNewDjangoClient():
    new_user = self.new_user()
    omeName = new_user.omeName.val
    new_django_client = self.new_django_client(omeName, omeName)
```

See also:

test_simple.py\(^3\) Example test class using the OMERO.web test library methods

---

\(^1\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroWeb/test/integration/weblibrary.py

\(^2\)https://docs.djangoproject.com/en/1.8/topics/testing/tools

\(^3\) 10.5. Running and writing tests 308
Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

11.1 OMERO Python language bindings

MOVIE: introduction to Blitz Gateway

In addition to the auto-generated Python libraries of the core OMERO Application Programming Interface, we have developed a more user-friendly Python module ‘Blitz Gateway’ that facilitates several aspects of working with the Python API, such as connection handling, object graph traversal and lazy loading.

This page gives you a large number of code samples to get you started. Then we describe a bit more about Blitz Gateway documentation.

The Python libraries are part of the server build and can be found under OMERO_HOME/lib/python. These include the core omero.model objects and services as well as the Blitz Gateway code (at OMERO_HOME/lib/python/omero/gateway/__init__.py).

To use OmeroPy, you will need to download the libraries (e.g. as part of the server package) and setup your PYTHONPATH to include them:

```bash
export OMERO_PREFIX=~/Desktop/OMERO.server-5.2.6-ice3x-byy # for example
export PYTHONPATH=$PYTHONPATH:$OMERO_PREFIX/lib/python
```

You will also need Ice libraries as described in the OMERO.server installation and an OMERO server to connect to, which must be the same major version, i.e. 5.2.x.

All the code examples below can be found at examples/Training/python.

If you want to run the examples, you will need to download and configure them to connect to your own server. E.g. `HOST = "localhost"` You can edit HOST, PORT, USERNAME and PASSWORD in the Parse_OMERO_Properties.py file and these values will be imported into the other scripts.

Then you can run the scripts:

```
$ python Connect_To_OMERO.py
```

If all goes well, you should be connected to your OMERO server and see some details of your session printed out.

All the following code examples can be downloaded and run in the same way. Some scripts will also need editing of other parameters, usually IDs from Projects, Datasets, Images etc. You can use the OMERO.insight or OMERO.web client to choose suitable data IDs before editing and running the code samples.

---

1 http://cvs.openmicroscopy.org.uk/snapshots/movies/omero-4-3/mov/BlitzGatewayIntro-4.3.mov
2 https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/examples/Training/python
11.1.1 Code samples

Connect to OMERO

```python
# Connect to the Python Blitz Gateway
# ==================================================================================================
# Make a simple connection to OMERO, printing details of the
# connection. See OmeroPy/Gateway for more info
conn = BlitzGateway(USERNAME, PASSWORD, host=HOST, port=PORT)
connected = conn.connect()

# Check if you are connected.
# ==================================================================================================
if not connected:
    import sys
    sys.stderr.write("Error: Connection not available, please check your user name and"
                     " password.\n")
    sys.exit(1)

# Using secure connection.
# ==================================================================================================
# By default, once we have logged in, data transfer is not encrypted
# (faster)
# To use a secure connection, call setSecure(True):

# conn.setSecure(True)  # <-------- Uncomment this

# Current session details
# ==================================================================================================
# By default, you will have logged into your 'current' group in OMERO. This
# can be changed by switching group in the OMERO.insight or OMERO.web clients.

user = conn.getUser()
print "Current user:"
print "   ID:" , user.getId()
print "   Username:" , user.getName()
print "   Full Name:" , user.getFullName()

print "Member of:"
for g in conn.getGroupsMemberOf():
    print "   ID:" , g.getName(), " Name:" , g.getId()

group = conn.getGroupFromContext()
print "Current group: ", group.getName()

# List the group owners and other members
owners, members = group.groupSummary()
print "   Group owners:"
for o in owners:
    print "      ID: %s %s Name: %s %s %s {" % (o.getId(), o.getOmeName(), o.getFullName())
print "      Group members:"
```

11.1. OMERO Python language bindings
for m in members:
    print "ID: %s Name: %s" % (m.getId(), m.getOmeName(), m.getFullName())

print "Owner of:"
for g in conn.listOwnedGroups():
    print "ID: ", g.getName(), " Name:",

# Added in OMERO 5.0
print "Admins:"
for exp in conn.getAdministrators():
    print "ID: %s Name: %s" % (exp.getId(), exp.getOmeName(), exp.getFullName())

# The 'context' of our current session
ctx = conn.getEventContext()
print ctx # for more info

# The 'context' of our current session
# When you are done, close the session to free up server resources.
conn._closeSession()

Read data

- Create a connection

conn = BlitzGateway(USERNAME, PASSWORD, host=HOST, port=PORT)
conn.connect()

- Configuration

imageId = 1
datasetId = 2
plateId = -1 # Don’t need to set this

def print_obj(obj, indent=0):
    ""
    Helper method to display info about OMERO objects.
    Not all objects will have a "name" or owner field.
    ""
    print """"%s%s Name:"%s (owner=%s)"""" % (  
        * * indent,  
        obj.OMERO_CLASS,  
        obj.getId(),  
        obj.getName(),  
        obj.getOwnerOmeName())

- List all Projects available to me, and their Datasets and Images:
# The only_owned=True parameter limits the Projects which are returned.
# If the parameter is omitted or the value is False, then all Projects
# visible in the current group are returned.
print "\nList Projects:
"
my_expId = conn.getUser().getId()
for project in conn.listProjects(my_expId):
    print_obj(project)
    for dataset in project.listChildren():
        print_obj(dataset, 2)
        for image in dataset.listChildren():
            print_obj(image, 4)

• Retrieve the datasets owned by the user currently logged in:

    # Here we create an omero.sys.ParametersI instance which we
    # can use to filter the results that are returned. If we did
    # not pass the params argument to getObjects, then all Datasets
    # in the current group would be returned.
print "\nList Datasets:
"
params = omero.sys.ParametersI()
params.exp(conn.getUser().getId())
# only show current user’s Datasets
datasets = conn.getObjects("Dataset", params=params)
for dataset in datasets:
    print_obj(dataset)

• Retrieve the images contained in a dataset:

    print "\nDataset:%s" % datasetId
    print "=" * 50
dataset = conn.getObject("Dataset", datasetId)
    print "\nImages in Dataset:%s", dataset.getName()
    for image in dataset.listChildren():
        print_obj(image)

• Retrieve an image by Image ID:

    image = conn.getObject("Image", imageId)
    print "\nImage:%s" % imageId
    print "=" * 50
    print image.getName(), image.getDescription()
    # Retrieve information about an image.
    print " X:", image.getSizeX()
    print " Y:", image.getSizeY()
    print " Z:", image.getSizeZ()
    print " C:", image.getSizeC()
    print " T:", image.getSizeT()
    # render the first timepoint, mid Z section
    z = image.getSizeZ() / 2
    t = 0
    renderedImage = image.renderImage(z, t)
    # renderedImage.show() # popup (use for debug only)
    # renderedImage.save("test.jpg") # save in the current folder
• Get Pixel Sizes for the above Image:

```python
sizeX = image.getPixelSizeX()  # E.g. 0.132
print "Pixel Size X:", sizeX
# Units support, new in OMERO 5.1.0
sizeXobj = image.getPixelSizeX(units=True)
print "Pixel Size X:", sizeXobj.getValue(), "(\$s)" % sizeXobj.getSymbol()
# To get the size with different units, E.g. Angstroms
sizeXang = image.getPixelSizeX(units="ANGSTROM")
print "Pixel Size X:", sizeXang.getValue(), "(\$s)" % sizeXang.getSymbol()
```

• Retrieve Screening data:

```python
print "\nList Screens:"
print "=* * 50
for screen in conn.getObjects("Screen"):
    print_obj(screen)
    for plate in screen.listChildren():
        print_obj(plate, 2)
        plateId = plate.getId()
```

• Retrieve Wells and Images within a Plate:

```python
if plateId != 0:
    print "\nPlate: \$s" % plateId
    print "=* * 50
    plate = conn.getObject("Plate", plateId)
    print "\nNumber of fields:", plate.getNumberOffFields()
    print "\nGrid size:", plate.getGridSize()
    print "\nWells in Plate:", plate.getName()
    for well in plate.listChildren():
        index = well.countWellSample()
        print "\nWell: ", well.row, well.column, " Fields: ", index
        for index in xrange(0, index):
            print "\nImage: ", \
            well.getImage(index).getName(), \
            well.getImage(index).getId()
```

• Close connection:

```python
# When you are done, close the session to free up server resources.
conn._closeSession()
```

**Groups and permissions**

• Create a connection

```python
conn = BlitzGateway(USERNAME, PASSWORD, host=HOST, port=PORT)
conn.connect()
```

• Configuration

```python
imageId = 1
```

• We are logged in to our ‘default’ group
group = conn.getGroupFromContext()
print "Current group: ", group.getName()

• Each group has defined Permissions set

group_perms = group.getDetails().getPermissions()
perm_string = str(group_perms)
permission_names = {
    'rwwr--': 'PRIVATE',
    'rwxr--': 'READ-ONLY',
    'rwxra-': 'READ-ANNOTATE',
    'rwrr--': 'READ-WRITE'}  # Not exposed in clients
print "Permissions: %s (%s)" % (permission_names[perm_string], perm_string)

• By default, any query applies to ALL data that we can access in our Current group.
This will be determined by group permissions e.g. in Read-Only or Read-annotate groups, this will include other users’ data - see Groups and permissions system.

projects = conn.listProjects()  # may include other users’ data
for p in projects:
    print p.getName(), "Owner: ", p.getDetails().getOwner().getFullName()

# Will return None if Image is not in current group
image = conn.getObject("Image", imageId)
print "Image: ", image

• In OMERO-4.4, we added ‘cross-group’ querying, use ‘-1’

c.conn.SERVICE_OPTS.setOmeroGroup('-1')
image = conn.getObject("Image", imageId)  # Will query across all my groups
print "Image: ", image,
if image is not None:
    print "Group: ", image.getDetails().getGroup().getName(),
    print image.details.group.id.val  # access groupId without loading group

• To query only a single group (not necessarily your ‘current’ group)

groupId = image.details.group.id.val
# This is how we ‘switch group’ in webclient
conn.SERVICE_OPTS.setOmeroGroup(groupId)
projects = conn.listProjects()
image = conn.getObject("Image", imageId)
print "Image: ", image,

• Close connection:

# When you are done, close the session to free up server resources.
conn._closeSession()

Raw data access

• Create a connection
conn = BlitzGateway(username, password, host=host, port=port)
conn.connect()

• Configuration

imageId = 27544

• Retrieve a given plane

# Use the pixelswrapper to retrieve the plane as a 2D numpy array. See [http://www.scipy.org/Tentative_NumPy_Tutorial]
# Numpy array can be used for various analysis routines
image = conn.getObject("Image", imageId)
sizeZ = image.getSizeZ()
sizeC = image.getSizeC()
sizeT = image.getSizeT()
z, t, c = 0, 0, 0
# first plane of the image
pixels = image.getPrimaryPixels()
plane = pixels.getPlane(z, c, t)
print "$\text{Plane at zct: }", z, c, t
print plane
print "$\text{shape: }", plane.shape
print "$\text{min: }", plane.min(), "$\text{ max: }", plane.max(), "$\text{ pixel type: }", plane.dtype.name"

• Retrieve a given stack

# Get a Z-stack of tiles. Using getTiles or getPlanes (see below) returns a generator of data (not all the data in hand) The RawPixelsStore is only opened once (not closed after each plane) Alternative is to use getPlane() or getTile() multiple times - slightly slower.
c, t = 0, 0
# First channel and timepoint
tile = (50, 50, 10, 10)
# list of [ (0,0,0,(x,y,w,h)), (1,0,0,(x,y,w,h)), (2,0,0,(x,y,w,h))... ]
zctList = [(iz, c, t, tile) for iz in range(sizeZ/2, sizeZ)]
print "$\text{Z stack of tiles:}"
planes = pixels.getTiles(zctList)
for i, p in enumerate(planes):
    print "$\text{Tile: }", zctList[i], "$\text{ min: }", p.min(), "$\text{ max: }", p.max(), "$\text{ sum: }", p.sum()"

• Retrieve a given hypercube

zctList = []
for z in range(sizeZ / 2, sizeZ):  # get the top half of the Z-stack
    for c in range(sizeC):  # all channels
        for t in range(sizeT):  # all time-points
            zctList.append((z, c, t))
print "$\text{Hyper stack of planes:}"
planes = pixels.getPlanes(zctList)
for i, p in enumerate(planes):
    print "$\text{plane zct: }", zctList[i], "$\text{ min: }", p.min(), "$\text{ max: }", p.max()

• Close connection:
# When you are done, close the session to free up server resources.
conn._closeSession()

## Write data

- **Create a connection**

  ```python
  conn = BlitzGateway(USERNAME, PASSWORD, host=HOST, port=PORT)
  conn.connect()
  ```

- **Configuration**

  ```python
  projectId = 2
  # Specify a local file. E.g. could be result of some analysis
  fileToUpload = "README.txt"  # This file should already exist
  ```

- **Create a new Dataset**

  ```python
  datasetObj = omero.model.DatasetI()
  datasetObj.setName(rstring("New Dataset"))
  datasetObj = conn.getUpdateService().saveAndReturnObject(datasetObj)
  datasetId = datasetObj.getId().getValue()
  print "New dataset, Id:", datasetId
  ```

- **Link to Project**

  ```python
  project = conn.getObject("Project", projectId)
  if project is None:
    import sys
    sys.stderr.write("Error: Object does not exist.\n")
    sys.exit(1)
  link = omero.model.ProjectDatasetLinkI()
  link.setParent(omero.model.ProjectI(project.getId(), False))
  link.setChild(datasetObj)
  conn.getUpdateService().saveObject(link)
  ```

- **Annotate Project with a new ‘tag’**

  ```python
  tagAnn = omero.gateway.TagAnnotationWrapper(conn)
  tagAnn.setValue("New Tag")
  tagAnn.save()
  project = conn.getObject("Project", projectId)
  project.linkAnnotation(tagAnn)
  ```

- **Added in OMERO 5.1: ‘Map’ annotations (list of key: value pairs)**

  ```python
  keyValueData = [["Drug Name", "Monastrol"],
                   ["Concentration", "5 mg/ml"]]
  mapAnn = omero.gateway.MapAnnotationWrapper(conn)
  # Use 'client' namespace to allow editing in Insight & web
  namespace = omero.constants.metadata.NSCLIENTMAPANNOTATION
  mapAnn.setNs(namespace)
  mapAnn.setValue(keyValueData)
  mapAnn.save()
  ```
```python
project = conn.getObject("Project", projectId)
# NB: only link a client map annotation to a single object
project.linkAnnotation(mapAnn)

• How to create a file annotation and link to a Dataset

dataset = conn.getObject("Dataset", datasetId)
# create the original file and file annotation (uploads the file etc.)
namespace = "imperial.training.demo"
print "\nCreating an OriginalFile and FileAnnotation"
fileAnn = conn.createCellAnnotFromLocalFile(
    fileToUpload, mimetype="text/plain", ns=namespace, desc=None)
print "\nAttaching FileAnnotation to Dataset: ", "File ID":", fileAnn.getId(), ", Name":", fileAnn.getName(), "Size":", fileAnn.getSize()
dataset.linkAnnotation(fileAnn)    # link it to dataset.

• Download a file annotation linked to a Dataset

# make a location to download the file. "download" folder.
path = os.path.join(os.path.dirname(__file__), "download")
if not os.path.exists(path):
    os.makedirs(path)

# Go through all the annotations on the Dataset. Download any file annotations we find.
print "\nAnnotations on Dataset:", dataset.getName()
for ann in dataset.listAnnotations():
    if isinstance(ann, omero.gateway.FileAnnotationWrapper):
        print "File ID":", ann.getFile().getId(), ann.getName(), "Size":", ann.getSize()

file_path = os.path.join(path, ann.getName())

f = open(str(file_path), 'w')
print "\nDownloading file to", file_path, "..."
try:
    for chunk in ann.getFileInChunks():
        f.write(chunk)
finally:
    f.close()
print "File downloaded!"

• Load all the file annotations with a given namespace

nsToInclude = [namespace]
sToExclude = []
metadataService = conn.getMetadataService()
annotations = metadataService.loadSpecifiedAnnotations(
    'omero.model.FileAnnotation', nsToInclude, nsToExclude, None)
for ann in annotations:
    print ann.getId().getValue(), ann.getName().val

• Get first annotation with specified namespace
```
ann = dataset.getAnnotation(namespace)
print "Found Annotation with namespace: ", ann.getNs()

• Close connection:

# When you are done, close the session to free up server resources.
conn._closeSession()

OMERO tables

• Create a connection

conn = BlitzGateway(USERNAME, PASSWORD, host=HOST, port=PORT)
conn.connect()

• Configuration

datasetId = 33

• Create a name for the Original File (should be unique)

from random import random
tablename = "TablesDemo:%s" % str(random())

coll = omero.grid.LongColumn('Uid', 'testLong', [])
col2 = omero.grid.StringColumn('MyStringColumnInit', '', 64, [])

columns = [coll, col2]

• Create and initialize a new table.

repositoryId = 1
table = conn.c.sf.sharedResources().newTable(repositoryId, tablename)
table.initialize(columns)

• Add data to the table.

ids = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
strings = ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine', 'ten']
data1 = omero.grid.LongColumn('Uid', 'test Long', ids)
data2 = omero.grid.StringColumn('MyStringColumn', '', 64, strings)
data = [data1, data2]
table.addData(data)
table.close()  # when we are done, close.

• Get the table as an original file...
orig_file = table.getOriginalFile()
orig_file_id = orig_file.id.val
# ...so you can attach this data to an object. E.g. Dataset
fileAnn = omero.model.FileAnnotationI()
# use unloaded OriginalFileI
fileAnn.setFile(omero.model.OriginalFileI(orig_file_id, False))
fileAnn = conn.getUpdateService().saveAndReturnObject(fileAnn)
link = omero.model.DatasetAnnotationLinkI()
link.setParent(omero.model.DatasetI(datasetId, False))
link.setChild(omero.model.FileAnnotationI(fileAnn.id.val, False))
conn.getUpdateService().saveAndReturnObject(link)

• Table API

See also:
OMERO Tables

openTable = conn.c.sf.sharedResources().openTable(orig_file)

print "Table Columns:"
for col in openTable.getHeaders():
    print ",", col.name
rowCount = openTable.getNumberOfRows()
print "Row count: ", rowCount

• Get data from every column of the specified rows

rowNumbers = [3, 5, 7]
print "\nGet All Data for rows: ", rowNumbers
data = openTable.readCoordinates(range(rowCount))
for col in data.columns:
    print "Data for Column: ", col.name
    for v in col.values:
        print ",", v

• Get data from specified columns of specified rows

colNumbers = [1]
start = 3
stop = 7
print "\nGet Data for cols: ", colNumbers, \
    " and between rows: ", start, ",", stop

data = openTable.read(colNumbers, start, stop)
for col in data.columns:
    print "Data for Column: ", col.name
    for v in col.values:
        print ",", v

• Query the table for rows where the ‘Uid’ is in a particular range

queryRows = openTable.getWhereList(" (Uid > 2) & (Uid <= 8) ", variables={}, start=0, stop=rowCount, step=0)

3http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/grid/Table.html
data = openTable.readCoordinates(queryRows)
for col in data.columns:
    print("Query Results for Column: ", col.name
    for v in col.values:
        print(" ", v
openTable.close()  # we're done

• In future, to get the table back from Original File

orig_table_file = conn.getObject("OriginalFile", attributes=({'name': tablename})  # if name is unique
savedTable = conn.c.sf.sharedResources().openTable(orig_table_file._obj)
print("Opened table with row-count:", savedTable.getNumberOfRows())

• Close connection:

# When you are done, close the session to free up server resources.
conn._closeSession()

ROIs

• Create a connection

conn = BlitzGateway(USERNAME, PASSWORD, host=HOST, port=PORT)
conn.connect()
updateService = conn.getUpdateService()

• Create ROI.

# We are using the core Python API and omero.model objects here, since ROIs
# are not yet supported in the Python Blitz Gateway.
#
# First we load our image and pick some parameters for shapes
x = 50
y = 200
width = 100
height = 50
image = conn.getObject("Image", imageId)
theZ = image.getSizeZ() / 2
theT = 0

# We have a helper function for creating an ROI and linking it to new shapes
def createROI(img, shapes):
    # create an ROI, link it to Image
    roi = omero.model.RoiI()
    # use the omero.model.ImageI that underlies the 'image' wrapper
    roi.setImage(img._obj)
    for shape in shapes:
        roi.addShape(shape)
    # Save the ROI (saves any linked shapes too)
    updateService.saveObject(roi)

# Another helper for generating the color integers for shapes
def rgbaToInt(red, green, blue, alpha=255):
Convert an R,G,B,A value to an int

```python
 RGBAInt = (alpha << 24) + (red << 16) + (green << 8) + blue
 if RGBAInt > (2**31 - 1):  # convert to signed 32-bit int
    RGBAInt = RGBAInt - 2**32
```

# create a rectangle shape (added to ROI below)
```python
print ("Adding a rectangle at theZ: $s, theT: $s, X: $s, Y: $s, width: $s," + " height: $s" % (theZ, theT, x, y, width, height))
rect = omero.model.RectangleI()
rect.x = rdouble(x)
rect.y = rdouble(y)
rect.width = rdouble(width)
rect.height = rdouble(height)
rect.theZ = rint(theZ)
rect.theT = rint(theT)
rect.textValue = rstring("test-Rectangle")
```

# create an Ellipse shape (added to ROI below)
```python
ellipse = omero.model.EllipseI()
eellipse.cx = rdouble(y)
eellipse.cy = rdouble(x)
eellipse.rx = rdouble(width)
eellipse.ry = rdouble(height)
eellipse.theZ = rint(theZ)
eellipse.theT = rint(theT)
eellipse.textValue = rstring("test-Ellipse")
```

# Create an ROI containing 2 shapes on same plane
# NB: OMERO.insight client doesn't support display
# of multiple shapes on a single plane.
# Therefore the ellipse is removed later (see below)
createROI(image, [rect, ellipse])

# create an ROI with single line shape
```python
line = omero.model.LineI()
line.x1 = rdouble(x)
line.x2 = rdouble(x+width)
line.y1 = rdouble(y)
line.y2 = rdouble(y+height)
line.theZ = rint(theZ)
line.theT = rint(theT)
line.textValue = rstring("test-Line")
createROI(image, [line])
```

```python
def create_mask(mask_bytes, bytes_per_pixel=1):
    if bytes_per_pixel == 2:
        divider = 16.0
        format_string = "H"  # Unsigned short
        byte_factor = 0.5
    elif bytes_per_pixel == 1:
        divider = 8.0
        format_string = "B"  # Unsigned char
        byte_factor = 1
    else:
        message = "Format $s not supported"
```

# 11.1. OMERO Python language bindings
raise ValueError(message)
steps = math.ceil(len(mask_bytes) / divider)
mask = []
for i in range(long(steps)):
    binary = mask_bytes[i * int(divider):i * int(divider) + int(divider)]
    format = str(int(byte_factor * len(binary))) + format_string
    binary = struct.unpack(format, binary)
    s = ''
    for bit in binary:
        s += str(bit)
    mask.append(int(s, 2))
return bytearray(mask)

mask_x = 50
mask_y = 50
mask_h = 100
mask_w = 100
# Create [0, 1] mask
mask_array = numpy.fromfunction(
    lambda x, y: (x * y) % 2, (mask_w, mask_h))
# Set correct number of bytes per value
mask_array = mask_array.astype(numpy.uint8)
# Convert the mask to bytes
mask_array = mask_array.tobytes()
# Pack the bytes to a bit mask
mask_packed = create_mask(mask_array, 1)

# Define mask’s fill color
mask_color = ColorHolder()
mask_color.setRed(255)
mask_color.setBlue(0)
mask_color.setGreen(0)
mask_color.setAlpha(100)

# Create an ROI with a single mask
mask = omero.model.MaskI()
mask.setTheC(rint(0))
mask.setTheZ(rint(0))
mask.setTheT(rint(0))
mask.setWidth(rdouble(mask_w))
mask.setHeight(rdouble(mask_h))
mask.setFillColor(rint(mask_color.getInt()))
mask.setTextValue(rstring("test-Mask"))
mask.setBytes(mask_packed)
createROI(image, [mask])

# create an ROI with single point shape
point = omero.model.PointI()
point.cx = rdouble(x)
point.cy = rdouble(y)
point.theZ = rint(theZ)
point.theT = rint(theT)
point.textValue = rstring("test-Point")
createROI(image, [point])
def pointsToString(points):
    """ Returns legacy format supported by Insight """
    points = ["%s,%s" % (p[0], p[1]) for p in points]
csv = " ", ".join(points)
    return "points1[%s] points2[%s] points3[%s]" % (csv, csv, csv)

# create an ROI with a single polygon, setting colors and lineWidth
polygon = omero.model.PolygonI()
polygon.theZ = rint(theZ)
polygon.theT = rint(theT)
polygon.fillColor = rint(rgbaToInt(255, 0, 255, 50))
polygon.strokeColor = rint(rgbaToInt(255, 255, 0))
polygon.strokeWidth = omero.model.LengthI(10, UnitsLength.PIXEL)
points = [[10, 20], [50, 150], [200, 200], [250, 75]]
createROI(image, [polygon])

- Retrieve ROIs linked to an Image.

roiService = conn.getRoiService()
result = roiService.findByImage(imageId, None)
for roi in result.rois:
    print "ROI: ID=%s", roi.getId().getValue()
    for s in roi.copyShapes():
        shape = {}
        shape['id'] = s.getId().getValue()
        shape['theT'] = s.getTheT().getValue()
        shape['theZ'] = s.getTheZ().getValue()
        if s.getTextValue() != None:
            shape['textValue'] = s.getTextValue().getValue()
        if type(s) == omero.model.RectangleI:
            shape['type'] = 'Rectangle'
            shape['x'] = s.getX().getValue()
            shape['y'] = s.getY().getValue()
            shape['width'] = s.getWidth().getValue()
            shape['height'] = s.getHeight().getValue()
        elif type(s) == omero.model.EllipseI:
            shape['type'] = 'Ellipse'
            shape['cx'] = s.getCx().getValue()
            shape['cy'] = s.getCy().getValue()
            shape['rx'] = s.getRx().getValue()
            shape['ry'] = s.getRy().getValue()
        elif type(s) == omero.model.PointI:
            shape['type'] = 'Point'
            shape['cx'] = s.getCx().getValue()
            shape['cy'] = s.getCy().getValue()
        elif type(s) == omero.model.LineI:
            shape['type'] = 'Line'
            shape['x1'] = s.getX1().getValue()
            shape['x2'] = s.getX2().getValue()
            shape['y1'] = s.getY1().getValue()
            shape['y2'] = s.getY2().getValue()
        elif type(s) == omero.model.MaskI:
            shape['type'] = 'Mask'
            shape['x'] = s.getX().getValue()
            shape['y'] = s.getY().getValue()
            shape['width'] = s.getWidth().getValue()
            shape['height'] = s.getHeight().getValue()
        elif type(s) in (omerod.model.LabelI, omero.model.PolygonI):
            print type(s), " Not supported by this code"
# Do some processing here, or just print:
print " Shape:",
for key, value in shape.items():
print " , key, value,
print "

• Remove shape from ROI

result = roiService.findByImage(imageId, None)
for roi in result.rois:
    for s in roi.copyShapes():
        # Find and remove the Shape we added above
        if s.getTextValue() and s.getTextValue().getValue() == "test-Ellipse":
            print "Removing Shape from ROI..."
            roi.removeShape(s)
            roi = updateService.saveAndReturnObject(roi)

• Close connection:

# When you are done, close the session to free up server resources.
conn._closeSession()

Delete data

• Create a connection

conn = BlitzGateway(USERNAME, PASSWORD, host=HOST, port=PORT)
conn.connect()

• Configuration

projectId = 507  # NB: This will be deleted!

• Load the Project

project = conn.getObject("Project", projectId)
if project is None:
    import sys
    sys.stderr.write("Error: Object does not exist.\n")
    sys.exit(1)

print "Project:", project.getName()

• Delete Project

# You can delete a number of objects of the same type at the same
# time. In this case ‘Project’. Use deleteChildren=True if you are
# deleting a Project and you want to delete Datasets and Images.
obj_ids = [projectId]
deleteChildren = False
handle = conn.deleteObjects(
    "Project", obj_ids, deleteAnns=True, deleteChildren=deleteChildren)

• Retrieve callback and wait until delete completes
# This is not necessary for the Delete to complete. Can be used if you want to know when delete is finished or if there were any errors.
cb = omero.callbacks.CmdCallbackI(conn.c, handle)
print "Deleting, please wait."
while not cb.block(500):
    print "..."
err = isinstance(cb.getResponse(), omero.cmd.ERR)
print "Error?", err
if err:
    print cb.getResponse()
    cb.close(True)  # close handle too

• Close connection:

# When you are done, close the session to free up server resources.
conn._closeSession()

**Render Images**

• Create a connection

conn = BlitzGateway(USERNAME, PASSWORD, host=HOST, port=PORT)
conn.connect()

• Configuration

imageId = 27544

• Get thumbnail

```
# Thumbnail is created using the current rendering settings on the image
image = conn.getObject("Image", imageId)
img_data = image.getThumbnail()
renderedThumb = Image.open(StringIO(img_data))
# renderedThumb.show()  # shows a pop-up
renderedThumb.save("thumbnail.jpg")
```

• Get current settings

```
print "Channel rendering settings:"
for ch in image.getChannels():
    # if no name, get emission wavelength or index
    print "Name: ", ch.getLabel()
    print "Color: ", ch.getColor().getHtml()
    print "Active: ", ch.isActive()
    print "Levels: ", ch.getWindowStart(), "-", ch.getWindowEnd()
print "isGreyscaleRenderingModel:", image.isGreyscaleRenderingModel()
print "Default Z/T positions:"
print "Z = \$s, T = \$s" % (image.getDefaultZ(), image.getDefaultT())
```

• Show the saved rendering settings on this image

```
print "Rendering Defs on Image:"
for rdef in image.getAllRenderingDefs():
```
```

# When you are done, close the session to free up server resources.
conn._closeSession()
```

---

### 11.1. OMERO Python language bindings

```

img_data = image.getThumbnail(rdefId=rdef['id'])
print "ID: %s (owner: %s %s)" % (rdef['id'], rdef['owner']["firstName", rdef['owner']["lastName"])

• Render each channel as a separate greyscale image

image.setGreyscaleRenderingModel()
sizeC = image.getSizeC()
z = image.getSizeZ() / 2
for c in range(1, sizeC + 1):  # Channel index starts at 1
    channels = [c]  # Turn on a single channel at a time
    image.setActiveChannels(channels)
    renderedImage = image.renderImage(z, t)
    renderedImage.show()  # popup (use for debug only)
    renderedImage.save("channel%s.jpg" % c)  # save in the current folder

• Turn 3 channels on, setting their colors

image.setColorRenderingModel()
channels = [1, 2, 3]
colorList = ['F00', None, 'FFFFFF00']  # do not change color of 2nd channel
image setActiveChannels(channels, colors=colorList)
image.setProjection('intmax')
renderedImage = image.renderImage(z, t)  # z and t are ignored for projections
renderedImage.show()
renderedImage.save("all_channels.jpg")
image.setProjection('normal')  # turn off projection

• Turn 2 channels on, setting levels of the first one

channels = [1, 2]
rangeList = [[100.0, 120.2], [None, None]]
image setActiveChannels(channels, windows=rangeList)
image.setDefaultZ(0)
image.setDefaultT(0)
# default compression is 0.9
renderedImage = image.renderImage(z=None, t=None, compression=0.9)
renderedImage.show()
renderedImage.save("two_channels.jpg")

• Save the current rendering settings & default Z/T

image.saveDefaults()

• Reset to settings at import time, and optionally save

image.resetDefaults(save=True)

• Close connection:

```
Create Image

• Create a connection

conn = BlitzGateway(USERNAME, PASSWORD, host=HOST, port=PORT)
conn.connect()

• Configuration

imageId = 27544  # This image must have at least 2 channels

• Create an image from scratch

# This example demonstrates the usage of the convenience method
# createImageFromNumpySeq() Here we create a multi-dimensional image from a
# hard-coded array of data.
from numpy import array, int8
import omero
sizeX, sizeY, sizeZ, sizeC, sizeT = 5, 4, 1, 2, 1
plane1 = array(
    [[0, 1, 2, 3, 4], [5, 6, 7, 8, 9], [0, 1, 2, 3, 4], [5, 6, 7, 8, 9]],
    dtype=int8)
plane2 = array(
    [[5, 6, 7, 8, 9], [0, 1, 2, 3, 4], [5, 6, 7, 8, 9], [0, 1, 2, 3, 4]],
    dtype=int8)
planes = [plane1, plane2]

def planeGen():
    """generator will yield planes""
    for p in planes:
        yield p

desc = "Image created from a hard-coded arrays"
i = conn.createImageFromNumpySeq("numpy image", sizeZ, sizeC, sizeT, description=desc, dataset=None)

print 'Created new Image:$s Name:"$s"' % (i.getId(), i.getName())

• Set the pixel size using units (added in 5.1.0)

  Lengths are specified by value and a unit enumeration Here we set the pixel size X and Y to be 9.8 Angstroms

from omero.model.enums import UnitsLength
# Re-load the image to avoid update conflicts
i = conn.getObject("Image", i.getId())
u = omero.model.LengthI(9.8, UnitsLength.ANGSTROM)
p = i.getPrimaryPixels()._obj
p.setPhysicalSizeX(u)
p.setPhysicalSizeY(u)
conn.getUpdateService().saveObject(p)

• Create an Image from an existing image
# We are going to create a new image by passing the method a ‘generator’ of 2D planes This will come from an existing image, by taking the average of 2 channels.

```python
zctList = []
image = conn.getObject('Image', imageId)
sizeZ, sizeC, sizeT = image.getSizeZ(), image.getSizeC(), image.getSizeT()
dataset = image.getParent()
pixels = image.getPrimaryPixels()
newSizeC = 1

def planeGen():
    """
    set up a generator of 2D numpy arrays.
    """
    for z in range(sizeZ): # all Z sections
        # Illustrative purposes only, since we only have 1 channel
        for c in range(newSizeC):
            for t in range(sizeT): # all time-points
                channel0 = pixels.getPlane(z, 0, t)
                channel1 = pixels.getPlane(z, 1, t)
                # Here we can manipulate the data in many different ways. As an example we are doing "average"
                newPlane = (channel0 + channel1) / 2
                print "newPlane for z,t": z, t, newPlane.dtype, \n                    newPlane.min(), newPlane.max()
                yield newPlane

desc =("Image created from Image ID: %s by averaging Channel 1 and Channel 2" % imageId)
i = conn.createImageFromNumpySeq(  
    planeGen(), "new image", sizeZ, newSizeC, sizeT, description=desc,  
    dataset=dataset)

• Close connection:

    # When you are done, close the session to free up server resources.
    conn._closeSession()
```

Filesets - added in OMERO 5.0

• Create a connection

```python
conn = BlitzGateway(USERNAME, PASSWORD, host=HOST, port=PORT)
conn.connect()
```

• Configuration

```python
imageId = 101
```

• Get the ‘Fileset’ for an Image
A Fileset is a collection of the original files imported to create an image or set of images in OMERO.

```python
image = conn.getObject("Image", imageId)
fileset = image.getFileset()  # will be None for pre-FS images
fsId = fileset.getId()
```

List all images that are in this fileset

```python
for fsImage in fileset.copyImages():
    print fsImage.getId(), fsImage.getName()
```

List original imported files

```python
for origFile in fileset.listFiles():
    name = origFile.getName()
    path = origFile.getPath()
    print path, name
```

- Get Original Imported Files directly from the image

```python
# this will include pre-FS data IF images were archived on import
print image.countImportedImageFiles()
# specifically count Fileset files
fileCount = image.countFilesetFiles()
```

List files

```python
if fileCount > 0:
    for origFile in image.getImportedImageFiles():
        name = origFile.getName()
        path = origFile.getPath()
        print path, name
```

- Can get the Fileset using conn.getObject()

```python
fileset = conn.getObject("Fileset", fsId)
```

- Close connection:

```python
# When you are done, close the session to free up server resources.
conn._closeSession()
```

Python OMERO.scripts

It is relatively straightforward to take the code samples above and re-use them in OMERO.scripts. This allows the code to be run on the OMERO server and called from either the OMERO.insight client or OMERO.web by any users of the server. See OMERO.scripts user guide.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

11.2 Blitz Gateway documentation

The Sphinx-generated documentation of methods provided by OMERO Gateway is available showing wrapper classes.

Specifically, the API for the ‘conn’ connection wrapper created above is here.

---


When working with OMERO model objects\(^6\) (omero.model.Image etc) the Gateway will wrap these objects in classes such as omero.gateway.ImageWrapper\(^7\) to handle object loading and hierarchy traversal. For example:

```python
>>> for p in conn.listProjects():  # Initially we just load Projects
    ... print p.getName()
    ... for dataset in p.listChildren():  # lazy-loading of Datasets here
    ...     print "", dataset.getName()

TestProject
Aurora-B
tiff stacks
newTimeStack
test
siRNAi
CENP
live-cell
survivin
```

### 11.2.1 Access to the OMERO API services

If you need access to API methods that are not provided by the gateway library, you can get hold of the OMERO Application Programming Interface.

**Note:** These services will always work with omero.model objects and not the gateway wrapper objects.

The gateway handles creation and reuse of the API services, so that new ones are not created unnecessarily. Services can be accessed using the methods of the underlying Service Factory\(^8\) with the Gateway handling reuse as needed. Stateless services (those retrieved with get... methods e.g. getQueryService\(^9\)) are always reused for each call, e.g. blitzon.getQueryService() whereas stateful services e.g. createRenderingEngine\(^10\) may be created each time.

Not all methods of the service factory are currently supported in the gateway. You can get an idea of the currently supported services by looking at the source code under the `_createProxies`\(^11\) method.

Example: ContainerService can load Projects and Datasets in a single call to server (no lazy loading)

```python
cs = conn.getContainerService()
projects = cs.loadContainerHierarchy("Project", None, None)
for p in projects:  # omero.model.Project
    print p.getName().getValue()  # need to `unwrap` rstring
    for d in p.linkedDatasetList():
        print d.getName().getValue()
```

### 11.2.2 Stateful services, reconnection, error handling etc

The Blitz gateway was designed for use in the OMERO.web framework framework and it is not expected that stateful services will be maintained on the client for significant time. There is various error-handling functionality in the Blitz gateway that will close existing services and recreate them in order to maintain a working connection. If this happens then any stateful services that you have on the client-side will become stale. We will attempt to document this a little better in due course, but our general advice is to create, use and close the stateful services in the shortest practicable time.

---

\(^6\)http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/model.html
\(^7\)http://downloads.openmicroscopy.org/latest/omero/api/python/omero/omero.gateway.html#omero.gateway.ImageWrapper
\(^8\)http://downloads.openmicroscopy.org/latest/omero/api/omero/api/ServiceFactory.html
\(^9\)http://downloads.openmicroscopy.org/latest/omero/api/python/omero/omero.api/ServiceFactory.html#getQueryService
\(^10\)http://downloads.openmicroscopy.org/latest/omero/api/omero/api/ServiceFactory.html#createRenderingEngine
\(^11\)http://downloads.openmicroscopy.org/latest/omero/api/python/omero/omero.gateway.html#omero.gateway._BlitzGateway._createProxies
11.2.3 Overwriting and extending omero.gateway classes

When working with omero.gateway\(^2\) or wrapper classes such as omero.gateway.ImageWrapper\(^3\) you might want to add your own functionality or customize an existing one. NB: Note the call to omero.gateway.refreshWrappers() to ensure that your subclasses are returned by calls to getObjects() For example:

```python
class MyBlitzGateway (omero.gateway.BlitzGateway):
    def __init__ (self, *args, **kwargs):
        super(MyBlitzGateway, self).__init__(*args, **kwargs)

        ...do something, e.g. add new field...
        self.new_field = 'foo'

    def connect (self, *args, **kwargs):
        rv = super(MyBlitzGateway, self).connect(*args,**kwargs)
        if rv:
            ...do something, e.g. modify new field...
            self.new_field = 'bla'

        return rv

omero.gateway.BlitzGateway = MyBlitzGateway

class MyBlitzObjectWrapper (object):
    annotation_counter = None

    def countAnnotations (self):
        ""
        Count on annotations linked to the object and set the value
        on the custom field 'annotation_counter'.
        ""

        @return Counter
        ""

        if self.annotation_counter is not None:
            return self.annotation_counter
        else:
            container = self._conn.getContainerService()
            m = container.getCollectionCount(self._obj.__class__.__name__,
                                                type(self._obj).ANNOTATIONLINKS, 
                                                [self._oid], None)
            if m[self._oid] > 0:
                self.annotation_counter = m[self._oid]
            return self.annotation_counter
        else:
            return None

class ImageWrapper (MyBlitzObjectWrapper, omero.gateway.ImageWrapper):
    ""
    omero_model_ImageI class wrapper overwrite omero.gateway.ImageWrapper
    and extends MyBlitzObjectWrapper.
    ""

    def __prepare__ (self, **kwargs):
        if kwargs.has_key('annotation_counter'):
            self.annotation_counte = kwargs['annotation_counter']

    omero.gateway.ImageWrapper = ImageWrapper

# IMPORTANT to update the map of wrappers for 'Image' etc. returned by getObjects("Image")
omero.gateway.refreshWrappers()
```

\(^2\)http://downloads.openmicroscopy.org/latest/omero/api/python/omero/omero.gateway.html
\(^3\)http://downloads.openmicroscopy.org/latest/omero/api/python/omero/omero.gateway.html#omero.gateway.ImageWrapper
This page provides some background information on the OMERO Python client ‘gateway’ (omero.gateway module) and describes work to improve the API.

The Blitz Gateway is a Python client-side library that facilitates working with the OMERO API, handling connection to the server, loading of data objects and providing convenience methods to access the data. It was originally designed as part of the OMERO.web framework framework, to provide connection and data retrieval services to various web clients. However, we have now decided to encourage its use for all access to the OMERO Python API.

11.2.4 Wrapper objects

The Gateway consists of a number of wrapper objects:

Connection wrapper

The BlitzGateway class (see API of development code) is a wrapper for the OMERO client and session objects. It provides various methods for connecting to the OMERO server, querying the status or context of the current connection and as a starting point for retrieving data objects from OMERO.

```python
from omero.gateway import *

conn = BlitzGateway("username", "password", host="localhost", port=4064)
conn.connect()

for p in conn.listProjects():
    print p.name
```

Model object wrappers

OMERO model objects, e.g. omero.model.Project, omero.model.Pixels etc. (see full list) are code-generated and mapped to the OMERO database schema. They are language agnostic and their data is in the form of omero.rtypes as described in about model objects).

```python
import omero
from omero.model import *
from omero.rtypes import rstring

p = omero.model.ProjectI()
p.name = rstring("My Project")   # attributes are all rtypes
print p.getName().getValue()     # getValue() to unwrap the rtype
print p.name.val                 # short-hand
```

To facilitate work in Python, particularly in web page templates, these Python model objects are wrapped in Blitz Object Wrappers. This hides the use of rtypes.

```python
import omero
from omero.model import *
from omero.rtypes import rstring

p = omero.model.ProjectI()
p.setName(rstring("Omero Model Project"))  # attributes are all rtypes
print p.getName().getValue()                 # getValue() to unwrap the rtype
print p.name.val                             # short-hand
```

---

14http://downloads.openmicroscopy.org/latest/omero/api/python/omero/omero.gateway.html#omero.gateway._BlitzGateway
15http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/model.html
from omero.gateway import *
project = ProjectWrapper(obj=p)  # wrap the model.object
project.setName("Project Wrapper")  # Don't need to use rtypes
print project.getName()
print project.name
print project._obj  # access the wrapped object with ._obj

These wrappers also have a reference to the BlitzGateway connection wrapper, so they can make calls to the server and load more data when needed (lazy loading).

E.g.

# connect as above
for p in conn.listProjects():
    print p.name
    for dataset in p.listChildren():  # lazy loading of datasets, wrapped in DatasetWrapper
        print "Dataset", d.name

Wrapper coverage

The OMERO data model has a large number of objects, not all of which are used by the OMERO.web framework. Therefore, the Blitz gateway (which was originally built for this framework) has not yet been extended to wrap every omero.model object with a specific Blitz Object Wrapper. The current list of object wrappers can be found in the omero.gateway module API\(^{16}\). As more functionality is provided by the Blitz Gateway, the coverage of object wrappers will increase accordingly.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

11.3 Command Line Interface as an OMERO development tool

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

11.3.1 Working with objects

The omero obj command allows to create and update OMERO objects. More information can be displayed using bin/omero obj -h.

Object creation

The omero obj new subcommand allows to create new objects:

$ bin/omero obj new Object field=value

where Object is the type of object to create, e.g. Dataset or ProjectDatasetLink and field/value is a valid key/value pair for the type of object. For example, the following command creates a new screen with a name and a description:

$ bin/omero obj new Screen name=Screen001 description="screen description"

\(^{16}\)http://downloads.openmicroscopy.org/latest/omero/api/python/omero/omero.gateway.html
Object update

The omero obj update subcommand allows to update existing objects:

```
$ bin/omero obj update Object:ID field=value
```

where Object:ID is the type and the ID of object to update, e.g. Image:1 or PlateDatasetLink:10 and field/value is a valid key/value pair to update for the specified object.

For example, the following command updates the existing screen of ID 2 with a name and a description:

```
$ bin/omero obj update Screen:2 name=Screen001 description="screen description"
```

Piping output

The output of each omero obj command is formatted as Object:ID so that the CLI commands can be redirected and piped together. For example, the following set of commands creates a dataset and a project and links them together:

```
$ dataset=$(bin/omero obj new Dataset name=dataset-1)
$ project=$(bin/omero obj new Project name=plate-1)
$ bin/omero obj new ProjectDatasetLink parent=$project child=$dataset
```

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

11.3.2 Extensions

Plugins can be written and put in the lib/python/omero/plugins directory. On execution, all plugins in that directory are registered with the CLI. Alternatively, the --path argument can be used to point to other plugin files or directories.

Thread-safety

The omero.cli.CLI should be considered not thread-safe. A single connection object is accessible from all plugins via self.ctx.conn(args), and it is assumed that changes to this object will only take place in the current thread. The CLI instance itself, however, can be passed between multiple threads, as long as only one accesses it sequentially, possibly via locking.

See also:

Extending OMERO.server Other extensions to OMERO

See also:

Command Line Interface as an OMERO client User documentation on the Command Line Interface

Command Line Interface as an OMERO admin tool System Administrator documentation for the Command Line Interface

Help for any specific CLI command can be displayed using the -h argument. See Command line help for more information.

11.3.3 General notes

- bin/omero will find its installation. Therefore, to install OMERO it is only necessary to unpack the bundle, and put bin/omero somewhere on your path.
- Any command can be produced by symlinking bin/omero to a file of the form “omero-command-arg1-arg2”. This is useful under /etc/rc.d to have a startup script.
- All commands respond to omero help.

11.3. Command Line Interface as an OMERO development tool
11.4 Omero Java language bindings

Using the Ice Java language mapping from ZeroC, Omero provides access to your data within an Omero.blitz server from Java code.

11.4.1 Writing client apps

To make use of the Omero Java API and interact with Omero.blitz from your code, a client application needs the Java bindings available on the classpath.

The required .jar files can be obtained in a number of ways:

- manually from the OME artifactory. All available artifacts and their POM files can be browsed using the Maven repository.
- using the Omero.java ZIP file downloaded from the Java section of the Omero download page. The libs directory can then be used on the Java classpath (or attached to a project in Eclipse).
- following the example in minimal-omero-client. Please make sure you are using the proper branch of the repository, as that influences the versions of dependencies defined in the Maven POM file.

11.4.2 Extended classpath

To access all the functionality available in omero_client.jar or to use the importer, you will need more jar files. To see all the current requirements, take a look at the builds on jenkins, or alternatively examine the dependencies in the ivy.xml files (e.g. components/insight/ivy.xml)

11.4.3 Connect to Omero

Connect to the server. Remember to close the session.

LoginCredentials cred = new LoginCredentials(userName, password, host, port);

// Alternative using args array:
// args = new String[]{"--omero.host=" + hostName, "--omero.port=" + port,
// "--omero.user=" + userName, "--omero.pass=" + password };
// LoginCredentials cred = new LoginCredentials(args);

Gateway gateway = new Gateway(simpleLogger);
ExperimenterData user = gateway.connect(cred);

// for every subsequent call to the server you’ll need the

Note: This documentation is for Omero 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. Omero 5.3 is expected before the end of 2016.
//SecurityContext for a certain group; in this case create
//a SecurityContext for the user’s default group.
SecurityContext ctx = new SecurityContext(user.getGroupId());

- Close connection. IMPORTANT

gateway.disconnect();

11.4.4 Read data

The BrowseFacility offers methods for browsing within the data hierarchy. A list of examples follows, indicating how to load Project, Dataset, Screen, etc.

- **Retrieve the projects** owned by the user currently logged in.

If a Project contains Datasets, the Datasets will automatically be loaded.

```java
BrowseFacility browse = gateway.getFacility(BrowseFacility.class);
Collection<ProjectData> projects = browse.getProjects(ctx);
Iterator<ProjectData> i = projects.iterator();
ProjectData project;
Set<DatasetData> datasets;
Iterator<DatasetData> j;
DatasetData dataset;
while (i.hasNext()) {
    project = i.next();
    String name = project.getName();
    long id = project.getId();
    datasets = project.getDatasets();
    j = datasets.iterator();
    while (j.hasNext()) {
        dataset = j.next();
        // Do something here
        // If images loaded.
        // dataset.getImages();
    }
}
```

- **Retrieve the Datasets** owned by the user currently logged in.

```java
BrowseFacility browse = gateway.getFacility(BrowseFacility.class);
Collection<DatasetData> datasets = browse.getDatasets(ctx);
Iterator<DatasetData> i = datasets.iterator();
DatasetData dataset;
Set<ImageData> images;
Iterator<ImageData> j;
ImageData image;
while (i.hasNext()) {
    dataset = i.next();
    images = dataset.getImages();
    j = images.iterator();
    while (j.hasNext()) {
        image = j.next();
        // Do something
    }
}
```
• **Retrieve the Images** contained in a Dataset.

```java
BrowseFacility browse = gateway.getFacility(BrowseFacility.class);
Collection<ImageData> images = browse.getImagesForDatasets(ctx, Arrays.asList(datasetId));
Iterator<ImageData> j = images.iterator();
ImageData image;
while (j.hasNext()) {
    image = j.next();
    // Do something
}
```

• **Retrieve an Image** if the identifier is known.

```java
BrowseFacility browse = gateway.getFacility(BrowseFacility.class);
ImageData image = browse.getImage(ctx, imageId);
```

• **Access information about the image** for example to draw it.

The model is as follows: Image-Pixels i.e. to access valuable data about the image you need to use the pixels object. We now only support one set of pixels per image (it used to be more!).

```java
PixelsData pixels = image.getDefaultPixels();
int sizeZ = pixels.getSizeZ(); // The number of z-sections.
int sizeT = pixels.getSizeT(); // The number of timepoints.
int sizeC = pixels.getSizeC(); // The number of channels.
int sizeX = pixels.getSizeX(); // The number of pixels along the X-axis.
int sizeY = pixels.getSizeY(); // The number of pixels along the Y-axis.
```

• **Retrieve Screening data** owned by the user currently logged in.

Note that the wells are not loaded.

```java
BrowseFacility browse = gateway.getFacility(BrowseFacility.class);
Collection<ScreenData> screens = browse.getScreens(ctx);
Iterator<ScreenData> i = screens.iterator();
ScreenData screen;
Set<PlateData> plates;
Iterator<PlateData> j;
PlateData plate;
while (i.hasNext()) {
    screen = i.next();
    plates = screen.getPlates();
    j = plates.iterator();
    while (j.hasNext()) {
        plate = j.next();
    }
}
```

• **Retrieve Wells within a Plate.**

Given a plate ID, load the wells.

```java
BrowseFacility browse = gateway.getFacility(BrowseFacility.class);
Collection<WellData> wells = browse.getWells(ctx, plateId);
Iterator<WellData> i = wells.iterator();
WellData well;
while (i.hasNext()) {
```
well = i.next();
//Do something
}

## 11.4.5 Raw data access

- **Retrieve a given plane.**

  This is useful when you need for example the pixels intensity.

  ```java
  RawDataFacility rdf = gateway.getFacility(RawDataFacility.class);
  PixelsData pixels = image.getDefaultPixels();
  int sizeZ = pixels.getSizeZ();
  int sizeT = pixels.getSizeT();
  int sizeC = pixels.getSizeC();

  Plane2D p;
  for (int z = 0; z < sizeZ; z++)
      for (int t = 0; t < sizeT; t++)
          for (int c = 0; c < sizeC; c++) {
              p = rdf.getPlane(ctx, pixels, z, t, c);
          }
  ```

- **Retrieve a given tile.**

  ```java
  RawDataFacility rdf = gateway.getFacility(RawDataFacility.class);

  PixelsData pixels = image.getDefaultPixels();
  int sizeZ = pixels.getSizeZ();
  int sizeT = pixels.getSizeT();
  int sizeC = pixels.getSizeC();
  int x = 0;
  int y = 0;
  int width = pixels.getSizeX()/2;
  int height = pixels.getSizeY()/2;

  Plane2D p;
  for (int z = 0; z < sizeZ; z++) {
      for (int t = 0; t < sizeT; t++) {
          for (int c = 0; c < sizeC; c++) {
              p = rdf.getTile(ctx, pixels, z, t, c, x, y, width, height);
          }
      }
  }
  ```

- **Retrieve a given stack.**

  This is useful when you need the pixels intensity.

  ```java
  PixelsData pixels = image.getDefaultPixels();
  int sizeT = pixels.getSizeT();
  int sizeC = pixels.getSizeC();
  long pixelsId = pixels.getId();
  RawPixelsStorePrx store = gateway.getPixelsStore(ctx);
  store.setPixelsId(pixelsId, false);
  for (int t = 0; t < sizeT; t++) {
      for (int c = 0; c < sizeC; c++) {
          byte[] plane = store.getStack(c, t);
          //Do something
      }
  }
  ```
• Retrieve a given hypercube.
This is useful when you need the pixels intensity.

PixelsData pixels = image.getDefaultPixels();
long pixelsId = pixels.getId();
// offset values in each dimension XYZCT
List<Integer> offset = new ArrayList<Integer>();
int n = 5;
for (int i = 0; i < n; i++) {
    offset.add(i, 0);
}
List<Integer> size = new ArrayList<Integer>();
size.add(pixels.getSizeX());
size.add(pixels.getSizeY());
size.add(pixels.getSizeZ());
size.add(pixels.getSizeC());
size.add(pixels.getSizeT());

// indicate the step in each direction, step = 1,
// will return values at index 0, 1, 2.
// step = 2, values at index 0, 2, 4 etc.
List<Integer> step = new ArrayList<Integer>();
for (int i = 0; i < n; i++) {
    step.add(i, 1);
}
RawPixelsStorePrx store = gateway.getPixelsStore(ctx);
store.setPixelsId(pixelsId, false);
byte[] values = store.getHypercube(offset, size, step);
// Do something
store.close();

11.4.6 Write data

• Create a dataset and link it to an existing project.

DataManagerFacility dm = gateway.getFacility(DataManagerFacility.class);

// Using IObject directly
Dataset dataset = new DatasetI();
dataset.setName(omero.rtypes.rstring("new Name 1"));
dataset.setDescription(omero.rtypes.rstring("new description 1"));

// Using the model object (recommended)
DatasetData datasetData = new DatasetData();
datasetData.setName("new Name 2");
datasetData.setDescription("new description 2");

ProjectDatasetLink link = new ProjectDatasetLinkI();
link.setChild(dataset);
link.setParent(new ProjectI(info.getProjectId(), false));
IObject r = dm.saveAndReturnObject(ctx, link);
// With model object
link = new ProjectDatasetLinkI();
link.setChild(datasetData.asDataset());
Import images into a dataset.

Using the Gateway:

```java
ImportCallback cb = new ImportCallback();
TransferFacility tf = gateway.getFacility(TransferFacility.class);

// the uploadImage method will automatically create a dataset with
// the same name as the directory, the image file is located in
tf.uploadImage(ctx, new File("/pathTo/image.dv"), cb);

// wait for the upload to finish
while (!cb.isFinished()) {
    try {
        Thread.sleep(250);
    } catch (InterruptedException e) {
    }
}
```

Using the Java API directly:

```java
String paths = new String[] {"/pathTo/image1.dv", "/pathTo/image2.dv"};

ImportConfig config = new ome.formats.importer.ImportConfig();

config.email.set("");
config.sendFiles.set(true);
config.sendReport.set(false);
config.contOnError.set(false);
config.debug.set(false);
config.hostname.set("localhost");
config.port.set(4064);
config.username.set("root");
config.password.set("omero");

// the imported image will go into 'orphaned images' unless
// you specify a particular existing dataset like this:
// config.targetClass.set("omero.model.Dataset");
// config.targetId.set(1L);

OMEROMetadataStoreClient store;
try {
    store = config.createStore();
    store.logVersionInfo(config.getIniVersionNumber());
    OMEROWrapper reader = new OMEROWrapper(config);
    ImportLibrary library = new ImportLibrary(store, reader);
    ErrorHandler handler = new ErrorHandler(config);
    library.addObserver(new LoggingImportMonitor());
    ImportCandidates candidates = new ImportCandidates(reader, paths, handler);
    reader.setMetadataOptions(new DefaultMetadataOptions(MetadataLevel.ALL));
    library.importCandidates(config, candidates);
    store.logout();
} catch (Exception e) {
    e.printStackTrace();
}
```
• Create a tag (tag annotation) and link it to an existing project.

```java
DataManagerFacility dm = gateway.getFacility(DataManagerFacility.class);

TagAnnotation tag = new TagAnnotationI();
tag.setTextValue(omero.rtypes.rstring("new tag 1"));
tag.setDescription(omero.rtypes.rstring("new tag 1"));

//Using the model object (recommended)
TagAnnotationData tagData = new TagAnnotationData("new tag 2");
tagData.setTagDescription("new tag 2");

ProjectAnnotationLink link = new ProjectAnnotationLinkI();
link.setChild(tag);
link.setParent(new ProjectI(info.getProjectId(), false));
IObject r = dm.saveAndReturnObject(ctx, link);

//With model object
link = new ProjectAnnotationLinkI();
link.setChild(tagData.asAnnotation());
link.setParent(new ProjectI(info.getProjectId(), false));
r = dm.saveAndReturnObject(ctx, link);
```

• Create a file annotation and link to an image.

To attach a file to an object e.g. an image, few objects need to be created:

1. an OriginalFile
2. a FileAnnotation
3. a link between the Image and the FileAnnotation.

```java
int INC = 262144;
DataManagerFacility dm = gateway.getFacility(DataManagerFacility.class);

//To retrieve the image see above.
File file = File.createTempFile("temp-file-name_", ".tmp");
String name = file.getName();
String absolutePath = file.getAbsolutePath();
String path = absolutePath.substring(0,
                          absolutePath.length()-name.length());

//create the original file object.
OriginalFile originalFile = new OriginalFileI();
originalFile.setName(omero.rtypes.rstring(name));
originalFile.setPath(omero.rtypes.rstring(path));
originalFile.setSize(omero.rtypes.rlong(file.length()));
final ChecksumAlgorithm checksumAlgorithm = new ChecksumAlgorithmI();
checksumAlgorithm.setValue(omero.rtypes.rstring(ChecksumAlgorithmSHA1160.value));
originalFile.setHasher(checksumAlgorithm);
originalFile.setMimeType(omero.rtypes.rstring(fileMimeType)); // or "application/octet-stream"
//Now we save the original file object
originalFile = (OriginalFile) dm.saveAndReturnObject(ctx, originalFile);

//Initialize the service to load the raw data
RawFileStorePrx rawFileStore = gateway.getRawFileService(ctx);
rawFileStore.setFileId(originalFile.getId().getValue());
//Open file and read stream.
FileInputStream stream = new FileInputStream(file);
long pos = 0;
int rlen;
byte[] buf = new byte[INC];
ByteBuffer bbuf;
while ((rlen = stream.read(buf)) > 0) {
    rawFileStore.write(buf, pos, rlen);
```
pos += rlen;
    bbuf = ByteBuffer.wrap(buf);
    bbuf.limit(rlen);
} stream.close();
originalFile = rawFileStore.save();
rawFileStore.close();

// now we have an original File in DB and raw data uploaded.
// We now need to link the Original file to the image using
// the File annotation object. That’s the way to do it.
FileAnnotation fa = new FileAnnotationI();
fa.setFile(originalFile);
fa.setDescription(omero.rtypes.rstring(description)); // The description set above e.g. PointsModel
fa.setNs(omero.rtypes.rstring(NAME_SPACE_TO_SET)); // The name space you have set to identify the file a

// save the file annotation.
fa = (FileAnnotation) dm.saveAndReturnObject(ctx, fa);

// now link the image and the annotation
ImageAnnotationLink link = new ImageAnnotationLinkI();
link.setChild(fa);
link.setParent(image.asImage());
// save the link back to the server.
link = (ImageAnnotationLink) dm.saveAndReturnObject(ctx, link);
// o attach to a Dataset use DatasetAnnotationLink;

    • Load all the file annotations with a given namespace.

long userId = gateway.getLoggedInUser().getId();
List<String> nsToInclude = new ArrayList<String>();
nsToInclude.add(NAME_SPACE_TO_SET);
List<String> nsToExclude = new ArrayList<String>();
ParametersI param = new ParametersI();
param.exp(omero.rtypes.rlong(userId)); // load the annotation for a given user.
IMetadataPrx proxy = gateway.getMetadataService(ctx);
List<Annotation> annotations = proxy.loadSpecifiedAnnotations(
    FileAnnotation.class.getName(), nsToInclude, nsToExclude, param);
// Do something with annotations.

    • Read the attachment.

First load the annotations, cf. above.

Iterator<Annotation> j = annotations.iterator();
Annotation annotation;
FileAnnotationData fa;
RawFileStorePrx store = null;
File file = File.createTempFile("temp-file-name_", ".tmp");
store = gateway.getRawFileService(ctx);
int index = 0;
FileOutputStream stream = new FileOutputStream(file);
OriginalFile of;
while (j.hasNext()) {
    annotation = j.next();
    if (annotation instanceof FileAnnotation && index == 0) {
        fa = new FileAnnotationData((FileAnnotation) annotation);
        // The id of the original file
        of = getOriginalFile(fa.getFileID());
        store.setIdId(fa.getFileID());
        int offset = 0;
        long size = of.getSize().getValue();
        pos += rlen;
        bbuf = ByteBuffer.wrap(buf);
        bbuf.limit(rlen);
    stream.close();
    originalFile = rawFileStore.save();
    rawFileStore.close();

// now we have an original File in DB and raw data uploaded.
// We now need to link the Original file to the image using
// the File annotation object. That’s the way to do it.
FileAnnotation fa = new FileAnnotationI();
fa.setFile(originalFile);
fa.setDescription(omero.rtypes.rstring(description)); // The description set above e.g. PointsModel
fa.setNs(omero.rtypes.rstring(NAME_SPACE_TO_SET)); // The name space you have set to identify the file a

// save the file annotation.
fa = (FileAnnotation) dm.saveAndReturnObject(ctx, fa);

// now link the image and the annotation
ImageAnnotationLink link = new ImageAnnotationLinkI();
link.setChild(fa);
link.setParent(image.asImage());
// save the link back to the server.
link = (ImageAnnotationLink) dm.saveAndReturnObject(ctx, link);
// o attach to a Dataset use DatasetAnnotationLink;

    • Load all the file annotations with a given namespace.

long userId = gateway.getLoggedInUser().getId();
List<String> nsToInclude = new ArrayList<String>();
nsToInclude.add(NAME_SPACE_TO_SET);
List<String> nsToExclude = new ArrayList<String>();
ParametersI param = new ParametersI();
param.exp(omero.rtypes.rlong(userId)); // load the annotation for a given user.
IMetadataPrx proxy = gateway.getMetadataService(ctx);
List<Annotation> annotations = proxy.loadSpecifiedAnnotations(
    FileAnnotation.class.getName(), nsToInclude, nsToExclude, param);
// Do something with annotations.

    • Read the attachment.

First load the annotations, cf. above.

Iterator<Annotation> j = annotations.iterator();
Annotation annotation;
FileAnnotationData fa;
RawFileStorePrx store = null;
File file = File.createTempFile("temp-file-name_", ".tmp");
store = gateway.getRawFileService(ctx);
int index = 0;
FileOutputStream stream = new FileOutputStream(file);
OriginalFile of;
while (j.hasNext()) {
    annotation = j.next();
    if (annotation instanceof FileAnnotation && index == 0) {
        fa = new FileAnnotationData((FileAnnotation) annotation);
        // The id of the original file
        of = getOriginalFile(fa.getFileID());
        store.setIdId(fa.getFileID());
        int offset = 0;
        long size = of.getSize().getValue();
        pos += rlen;
        bbuf = ByteBuffer.wrap(buf);
        bbuf.limit(rlen);
    stream.close();
    originalFile = rawFileStore.save();
    rawFileStore.close();

// now we have an original File in DB and raw data uploaded.
// We now need to link the Original file to the image using
// the File annotation object. That’s the way to do it.
FileAnnotation fa = new FileAnnotationI();
fa.setFile(originalFile);
fa.setDescription(omero.rtypes.rstring(description)); // The description set above e.g. PointsModel
fa.setNs(omero.rtypes.rstring(NAME_SPACE_TO_SET)); // The name space you have set to identify the file a

// save the file annotation.
fa = (FileAnnotation) dm.saveAndReturnObject(ctx, fa);

// now link the image and the annotation
ImageAnnotationLink link = new ImageAnnotationLinkI();
link.setChild(fa);
link.setParent(image.asImage());
// save the link back to the server.
link = (ImageAnnotationLink) dm.saveAndReturnObject(ctx, link);
// o attach to a Dataset use DatasetAnnotationLink;

    • Load all the file annotations with a given namespace.

long userId = gateway.getLoggedInUser().getId();
List<String> nsToInclude = new ArrayList<String>();
nsToInclude.add(NAME_SPACE_TO_SET);
List<String> nsToExclude = new ArrayList<String>();
ParametersI param = new ParametersI();
param.exp(omero.rtypes.rlong(userId)); // load the annotation for a given user.
IMetadataPrx proxy = gateway.getMetadataService(ctx);
List<Annotation> annotations = proxy.loadSpecifiedAnnotations(
    FileAnnotation.class.getName(), nsToInclude, nsToExclude, param);
// Do something with annotations.

    • Read the attachment.

First load the annotations, cf. above.

Iterator<Annotation> j = annotations.iterator();
Annotation annotation;
FileAnnotationData fa;
RawFileStorePrx store = null;
File file = File.createTempFile("temp-file-name_", ".tmp");
store = gateway.getRawFileService(ctx);
int index = 0;
FileOutputStream stream = new FileOutputStream(file);
OriginalFile of;
while (j.hasNext()) {
    annotation = j.next();
    if (annotation instanceof FileAnnotation && index == 0) {
        fa = new FileAnnotationData((FileAnnotation) annotation);
        // The id of the original file
        of = getOriginalFile(fa.getFileID());
        store.setIdId(fa.getFileID());
        int offset = 0;
        long size = of.getSize().getValue();

11.4.7 How to use OMERO tables

- Create a table.

In the following example, we create a table with 2 columns.

```java
/**
 * Creates a number of empty rows.
 *
 * @param rows The number of rows.
 * @return See above.
 */
private Column[] createColumns(int rows) {
    Column[] newColumns = new Column[2];
    newColumns[0] = new LongColumn("Uid", "", new long[rows]);
    return newColumns;
}

int rows = 1;
String name = UUID.randomUUID().toString();
Column[] columns = createColumns(rows);

//create a new table.
SharedResourcesPrx store = gateway.getSharedResources(ctx);
TablePrx table = store.newTable(1, name);

//initialize the table
table.initialize(columns);

//add data to the table.
rows = 2;
Column[] newRow = createColumns(rows);

LongColumn uids = (LongColumn) newRow[0];
LongColumn myLongs = (LongColumn) newRow[1];
for (int i = 0; i < rows; i++) {
    uids.values[i] = i;
    myLongs.values[i] = i;
}
table.addData(newRow);

OriginalFile file = table.getOriginalFile(); //if you need to interact with the table
```
• Read the contents of the table.

```java
file = new OriginalFileI(file.getId(), false);
table = store.openTable(file);

//read headers
Column[] cols = table.getHeaders();
for (int i = 0; i < cols.length; i++) {
    String colName = cols[i].name;
}

//Depending on size of table, you may only want to read some blocks.
long[] columnsToRead = new long[cols.length];
for (int i = 0; i < cols.length; i++) {
    columnsToRead[i] = i;
}

//The number of columns we wish to read.
long[] rowSubset = new long[(int) (table.getNumberOfRows()-1)];
for (int j = 0; j < rowSubset.length; j++) {
    rowSubset[j] = j;
}
Data data = table.slice(columnsToRead, rowSubset); // read the data.
for (int j = 0; j < cols.length; j++) {
    Column c = cols[j];
}
table.close();
```

### 11.4.8 ROIs

To learn about the model see the [ROI Model documentation](http://www.openmicroscopy.org/site/support/ome-model/developers/roi.html). Note that annotations can be linked to ROI or shape.

• Create ROI.

In this example, we create an ROI with a rectangular shape and attach it to an image.

```java
DataManagerFacility dm = gateway.getFacility(DataManagerFacility.class);
ROIFacility roifac = gateway.getFacility(ROIFacility.class);

//To retrieve the image see above.
ROIData data = new ROIData();
data.setImage(image);
//Create a rectangle.
RectangleData rectangle = new RectangleData(10, 10, 10, 10);
rectangle.setZ(0);
rectangle.setT(0);
data.addShapeData(rectangle);

//Create an ellipse.
EllipseData ellipse = new Ellipse(10, 10, 10, 10);
ellipse.setZ(0);
ellipse.setT(0);
ellipse.setText("ellipse text");
data.addShapeData(ellipse);

// Save ROI and shape
ROIData roiData = roifac.saveROIs(ctx, image.getId(), Arrays.asList(data)).iterator().next();
```

25http://www.openmicroscopy.org/site/support/ome-model/developers/roi.html
Retrieve ROIs linked to an Image.

```java
ROIFacility roifac = gateway.getFacility(ROIFacility.class);

// Retrieve the roi linked to an image
List<ROIResult> roiresults = roifac.loadROIs(ctx, image.getId());
ROIResult r = roiresults.iterator().next();
if (r == null) return;
List<Roi> rois = r.rois;
List<Shape> list;
Iterator<Roi> j = rois.iterator();
while (j.hasNext()) {
    roi = j.next();
    list = roi.copyShapes();
    // Do something
}
```

Remove a shape from ROI.

```java
DataManagerFacility dm = gateway.getFacility(DataManagerFacility.class);
ROIFacility roifac = gateway.getFacility(ROIFacility.class);

// Retrieve the roi linked to an image
List<ROIResult> roiresults = roifac.loadROIs(ctx, image.getId());
ROIResult r = roiresults.iterator().next();
List<Roi> rois = r.rois;
List<Shape> list;
Iterator<Roi> j = rois.iterator();
while (j.hasNext()) {
    roi = j.next();
    list = roi.copyShapes();
    // remove the first shape.
    if (list.size() > 0) {
        roi.removeShape(list.get(0));
        // update the roi.
        dm.saveAndReturnObject(ctx, roi).saveAndReturnObject(roi);
    }
}
```
11.4.9 Delete data

It is possible to delete Projects, datasets, images, ROIs etc. and objects linked to them depending on the specified options (see Deleting in OMERO).

- Delete Image.

In the following example, we create an image and delete it.

```java
DataManagerFacility dm = gateway.getFacility(DataManagerFacility.class);

// First create an image.
ImageData image = new ImageData();
image.setName("image1");
image.setDescription("descriptionImage1");
IObject object = dm.saveAndReturnObject(ctx, image.asIObject());
Response rsp = dm.deleteObject(ctx, object);
```

11.4.10 Render Images

- Initialize the rendering engine and render an image.

```java
PixelsData pixels = image.getDefaultPixels();
long pixelsId = pixels.getId();
RenderingEnginePrx proxy = null;
proxy = gateway.getRenderingService(ctx, pixelsId);
proxy.lookupPixels(pixelsId);
if (!(proxy.lookupRenderingDef(pixelsId))) {
    proxy.resetDefaultSettings(true);
    proxy.lookupRenderingDef(pixelsId);
}
proxy.load();
// Now can interact with the rendering engine.
proxy.getActive(0, Boolean.valueOf(false));
PlaneDef pDef = new PlaneDef();
pDef.z = 0;
pDef.t = 0;
pDef.slice = omero.romio.XY.value;
// render the data uncompressed.
int[] uncompressed = proxy.renderAsPackedInt(pDef);
byte[] compressed = proxy.renderCompressed(pDef);
// Create a buffered image
ByteArrayInputStream stream = new ByteArrayInputStream(compressed);
BufferedImage image = ImageIO.read(stream);
proxy.close();
```

- Retrieve thumbnails.

```java
ThumbnailStorePrx store = gateway.getThumbnailService(ctx);
PixelsData pixels = image.getDefaultPixels();
store.setPixelsId(pixels.getId());
// retrieve a 96x96 thumbnail.
byte[] array = store.getThumbnail(
```
omero.rtypes.rint(96), omero.rtypes.rint(96));
ByteArrayInputStream stream = new ByteArrayInputStream(array);
//Create a buffered image to display
ImageIO.read(stream);
store.close();

11.4.11 Create Image

The following example shows how to create an Image from an Image already in OMERO. Similar approach can be applied when uploading an image.

//See above how to load an image.
PixelsData pixels = image.getDefaultPixels();
int sizeZ = pixels.getSizeZ();
int sizeT = pixels.getSizeT();
int sizeC = pixels.getSizeC();
int sizeX = pixels.getSizeX();
int sizeY = pixels.getSizeY();
long pixelsId = pixels.getId();

//Read the pixels from the source image.
RawPixelsStorePrx store = gateway.getPixelsStore(ctx);
store.setPixelsId(pixelsId, false);
List<byte[]> planes = new ArrayList<byte[]>();
for (int z = 0; z < sizeZ; z++) {
    for (int t = 0; t < sizeT; t++) {
        planes.add(store.getPlane(z, 0, t));
    }
}
//Better to close to free space.
store.close();

//Now we are going to create the new image.
IPixelsPrx proxy = gateway.getPixelsService(ctx);

//Search for PixelsType object matching the source image.
List<IObject> l = proxy.getAllEnumerations(PixelsType.class.getName());
Iterator<IObject> i = l.iterator();
PixelsType type = null;
String original = pixels.getPixelType();
while (i.hasNext()) {
    PixelsType o = (PixelsType) i.next();
    String value = o.getValue().getValue();
    if (value.equals(original)) {
        type = o;
        break;
    }
}
if (type == null)
    throw new Exception("Pixels Type not valid.");

//Create new image.
String name = "newImageFrom"+image.getId();
RLong idNew = proxy.createImage(sizeX, sizeY, sizeZ, sizeT, Arrays.asList(0), type, name,
    "From Image ID: "+image.getId());
if (idNew == null)
    throw new Exception("New image could not be created.");
IContainerPrx proxyCS = entryUnencrypted.getContainerService();

11.4. OMERO Java language bindings 347
List<Image> results = proxyCS.getImages(Image.class.getName(), Arrays.asList(idNew.getValue()), new ParametersI());
ImageData newImage = new ImageData(results.get(0));

// Link the new image and the dataset hosting the source image.
DatasetImageLink link = new DatasetImageLinkI();
link.setParent(new DatasetI(datasetId, false));
link.setChild(new ImageI(newImage.getId(), false));
gateway.getUpdateService(ctx).saveAndReturnObject(link);

// Write the data.
store = gateway.getPixelsStore(ctx);
store.setPixelsId(newImage.getDefaultPixels().getId(), false);
int index = 0;
for (int z = 0; z < sizeZ; z++) {
  for (int t = 0; t < sizeT; t++) {
    store.setPlane(planes.get(index++), z, 0, t);
  }
}

// Save the data.
store.save();
store.close();

11.4.12 Further information

For the details behind writing, configuring, and executing a client, please see Working with OMERO.

See also:

ZeroC26, OMERO.grid, OmeroTools, OMERO Application Programming Interface

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

11.5 OMERO MATLAB language bindings

See Developing OMERO clients and OME-Remote Objects, for an introduction to Object.

11.5.1 Installing the OMERO.matlab toolbox

- Download the latest released version from the download page27.
- Unzip the directory anywhere on your system.
- In MATLAB, move to the newly unzipped directory and run loadOmero;.
- The MATLAB files are now on your path, and the necessary jars are on your Java classpath. You can change directories and still have access to OMERO.

Once OMERO.matlab is installed, the typical workflow is:
1. Creating a connection
2. Keeping your session alive

26https://zeroc.com
27http://downloads.openmicroscopy.org/latest/omero/
3. **Creating an unencrypted session** (optional)

4. Do some work (load objects, work with them, upload to the server…)

5. **Closing your connection**

6. **Unloading OMERO** (optional)

As a quickstart example, the following lines create a secure connection to a server, read a series of images and close the connection.

```matlab
client = loadOmero(servername, port);
session = client.createSession(user, password);
client.enableKeepAlive(60);
images = getImages(session, ids);
client.closeSession();
```

Examples of usage of the OMERO.matlab toolbox are provided in the training examples directory.

### 11.5.2 Configuring the OMERO.matlab connection

#### Creating a connection

As described under *Working with OMERO*, there are several ways to configure your connection to an OMERO server. OMERO.matlab comes with a few conveniences for making this work.

If you run `client = loadOmero();` (i.e. `loadOmero` with an output argument), then OMERO.matlab will try to configure the `omero.client` object for you. First, it checks the `ICE_CONFIG` environment variable. If set, it will let the `omero.client` constructor initialize itself. Otherwise, it looks for the file `ice.config` in the current directory. The OMERO.matlab toolbox comes with a default `ice.config` file pointing at `localhost`. To use this configuration file, you should replace `localhost` by your server address.

Alternatively, you can pass the same parameters to `loadOmero` that you would pass to `omero.client`:

```matlab
>> omero_client_1 = loadOmero('localhost');
>> omero_client_2 = omero.client('localhost');
```

Or, if you want a session created directly, the following are equivalent:

```matlab
>> [client1, session1] = loadOmero('localhost');
>> client2 = loadOmero('localhost');
>> session2 = client2.createSession()
```

#### Keeping your session alive

For executing any long running task, you will need a background thread which keeps your session alive. If you are familiar with MATLAB Timers you can use `omeroKeepAlive.m` directly or modify it to your liking.

```matlab
>> [c,s] = loadOmero;  
>> t = omeroKeepAlive(c); % Create a 60-second timer and starts it  
>> -  
>> delete(t);  % Disable the keep-alive
```

Alternatively, you can use the Java-based `enableKeepAlive` method, but it is not configurable from within MATLAB:

---

28[https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/examples/Training/matlab](https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/examples/Training/matlab)

29[https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/omeroKeepAlive.m](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/omeroKeepAlive.m)
c.enableKeepAlive(60); % Call session.keepAlive() every 60 seconds
c.closeSession(); % Close session to end the keep-alive

### Working in a different group

Each session is created within a given context, defining not only the session user but also the session group. The session context can be retrieved using the administration service:

```java
eventContext = s.getAdminService().getEventContext();
groupId = eventContext.groupId;
```

Most read and write operations described below are performed in the context of the session group when using the default parameters. Since OMERO 5.1.4, it is possible to specify a different context than the session group for reading and writing data using the `group` parameter/key value in the OMERO.matlab functions. Retrieving objects by identifiers is also done across all groups by default.

See also:

- OMERO permissions history, querying and usage
- Developer documentation about the OMERO permissions system

### Creating an unencrypted session

Once a session has been created, if you want to speed up the data transfer, you can create and use an unencrypted session as:

```java
unsecureClient = client.createClient(false);
sessionUnencrypted = unsecureClient.getSession();
```

### Closing your connection

When you are done with OMERO, it is critical that you close your connection to save resources:

```java
client.closeSession();
clear client1;
clear session1;
```

If you created an unencrypted session, you will need to close the unsecure session as well:

```java
client.closeSession();
unsecureClient.closeSession();
```

### Unloading OMERO

Then if you would like, you can unload OMERO as well:

```java
unloadOmero();
```

You may see the following warning when unloading OMERO:
OMERO Documentation, Release 5.2.6

>> unloadOmero()
Warning: Objects of omero/client class exist - not clearing java
  > In javaclasspath>doclear at 377
    In javaclasspath>local_javapath at 194
    In javaclasspath at 105
    In javarmpath at 48
    In unloadOmero at 75

While unloading OMERO, found java objects left in workspace.
Please remove with 'clear <name>' and then run 'unloadOmero'
again. Printing all objects...

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Bytes</th>
<th>Class</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>1x1</td>
<td></td>
<td>omero.client</td>
<td></td>
</tr>
</tbody>
</table>

Closing session(s) for 1 found client(s): c

This means that there is still an OMERO.matlab object in your workspace. If not listed, use whos to find such objects, and clear
to remove them. After that, run unloadOmero() again:

>> clear c
>> unloadOmero()

**Warning:** You should also unload OMERO before installing a new version of OMERO.matlab or calling loadOmero
again.

If you need to create another session without unloading/loading OMERO again, use the omero.client object directly:

>> [c,s] = loadOmero(arg1,arg2);
>> c = omero.client(arg3,arg4);
>> s = c.createSession();

### 11.5.3 Reading data

The IContainer service provides methods to load the data management hierarchy in OMERO – projects, datasets… A list of
eamples follows indicating how to load projects, datasets, screens…

- **Projects**

  The projects owned by the session user in the context of the session group can be retrieved using the `getProjects` function:

  ```matlab
  projects = getProjects(session)
  ```

  If the project identifiers are known, they can be retrieved independently of their owner or group using:

  ```matlab
  projects = getProjects(session, ids)
  ```

  If the projects contain datasets, the datasets will automatically be loaded:

  ```matlab
  ```
for \( j = 1 \): numel(projects)
    datasetsList = projects(j).linkedDatasetList;
    for \( i = 0 \) : datasetsList.size()-1,
        d = datasetsList.get(i);
    end
end

If the datasets contain images, the images are not automatically loaded. To load the whole graph (projects, datasets, images), pass \texttt{true} as an optional argument:

\[
\text{myLoadedProjects} = \text{getProjects(session, true)}
\]
\[
\text{loadedProjects} = \text{getProjects(session, ids, true)}
\]
\[
\text{imageList} = \text{loadedProjects}(1).\text{linkedDatasetList.get(0).linkedImageList};
\]

\textbf{Warning:} Loading the entire projects/datasets/images graph can be time-consuming and memory-consuming depending on the amount of data.

To return the orphaned datasets as well as the projects, you can query the second output argument of \texttt{getProjects}\textsuperscript{31}:

\[
\text{[projects, orphanedDatasets]} = \text{getProjects(session)}
\]

To filter projects by owner, use the \texttt{owner} parameter/key value. A value of -1 means projects are retrieved independently of their owner:

\texttt{\% Returns all projects owned by the specified user in the context of the session group}
\texttt{\% projects = getProjects(session, ‘owner’, ownerId);}  
\texttt{\% Returns all projects with the input identifiers owned by the specified user}
\texttt{\% projects = getProjects(session, ids, ‘owner’, ownerId);}  
\texttt{\% Returns all projects owned by any user in the context of the session group}
\texttt{\% projects = getProjects(session, ‘owner’, -1);}  

To filter projects by group, use the \texttt{group} parameter/key value. A value of -1 means projects are retrieved independently of their group:

\texttt{\% Returns all projects owned by the session user in the specified group}
\texttt{\% projects = getProjects(session, ‘group’, groupId);}  
\texttt{\% Returns all projects with the input identifiers in the specified group}
\texttt{\% projects = getProjects(session, ids, ‘group’, groupId);}  
\texttt{\% Returns all projects owned by the session user across groups}
\texttt{\% projects = getProjects(session, ‘group’, -1);}  

- Datasets

The datasets owned by the session user in the context of the session group can be retrieved using the \texttt{getDatasets}\textsuperscript{32} function:

\[
\text{datasets} = \text{getDatasets(session)}
\]

If the dataset identifiers are known, they can be retrieved independently of their owner or group using:

\textsuperscript{31}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/io/getProjects.m  
\textsuperscript{32}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/io/getDatasets.m
datasets = getDatasets(session, ids)

If the datasets contain images, the images are not automatically loaded. To load the whole graph (datasets, images), pass true as an optional argument:

loadedDatasets = getDatasets(session, ids, true);
imageList = loadedDatasets(1).linkedImageList;

**Warning:** Loading the entire datasets/images graph can be time-consuming and memory-consuming depending on the amount of data.

To filter datasets by owner, use the `owner` parameter/key value. A value of -1 means datasets are retrieved independently of their owner:

```
% Returns all datasets owned by the specified user in the context of the session group
datasets = getDatasets(session, 'owner', ownerId);
% Returns all datasets with the input identifiers owned by the specified user
datasets = getDatasets(session, ids, 'owner', ownerId);
% Returns all datasets owned by any user in the context of the session group
datasets = getDatasets(session, 'owner', -1);
```

To filter datasets by group, use the `group` parameter/key value. A value of -1 means datasets are retrieved independently of their group:

```
% Returns all datasets owned by the session user in the specified group
datasets = getDatasets(session, 'group', groupId);
% Returns all datasets with the input identifiers in the specified group
datasets = getDatasets(session, ids, 'group', groupId);
% Returns all datasets owned by the session user across groups
datasets = getDatasets(session, 'group', -1);
```

• **Images**

The images owned by the session user in the context of the session group can be retrieved using the `getImages` function:

```
images = getImages(session)
```

If the image identifiers are known, they can be retrieved independently of their owner or group using:

```
images = getImages(session, ids)
```

All the images contained in a subset of datasets of known identifiers `datasetsIds` can be returned independently of their owner or group using:

```
datasetImages = getImages(session, 'dataset', datasetsIds)
```

All the images contained in all the datasets under a subset of projects of known identifiers `projectIds` can be returned independently of their owner or group using:

```
https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/io/getImages.m
```

11.5. OMERO MATLAB language bindings
projectImages = getImages(session, 'project', projectIds)

To filter images by owner, use the owner parameter/key value. A value of -1 means images are retrieved independently of their owner:

% Returns all images owned by the specified user in the context of the session group
images = getImages(session, 'owner', ownerId);
% Returns all images with the input identifiers owned by the specified user
images = getImages(session, ids, 'owner', ownerId);
% Returns all images owned by any user in the context of the session group
images = getImages(session, 'owner', -1);

To filter images by group, use the group parameter/key value. A value of -1 means images are retrieved independently of their group:

% Returns all images owned by the session user in the specified group
images = getImages(session, 'group', groupId);
% Returns all images with the input identifiers in the specified group
images = getImages(session, ids, 'group', groupId);
% Returns all images owned by the session user across groups
images = getImages(session, 'group', -1);

The Image-Pixels model implies you need to use the Pixels objects to access valuable data about the Image:

pixels = image.getPrimaryPixels();
sizeZ = pixels.getSizeZ().getValue(); % The number of z-sections.
sizeT = pixels.getSizeT().getValue(); % The number of timepoints.
sizeC = pixels.getSizeC().getValue(); % The number of channels.
sizeX = pixels.getSizeX().getValue(); % The number of pixels along the X-axis.
sizeY = pixels.getSizeY().getValue(); % The number of pixels along the Y-axis.

• Screens

The screens owned by the session user in the context of the session group can be retrieved using the getScreens function:

screens = getScreens(session)

If the screen identifiers are known, they can be retrieved independently of their owner or group using:

screens = getScreens(session, ids)

Note that the wells are not loaded. The plate objects can be accessed using:

for j = 1 : numel(screens),
    platesList = screens(j).linkedPlateList;
    for i = 0:platesList.size()-1,
        plate = platesList.get(i);
        plateAcquisitionList = plate.copyPlateAcquisitions();
        for k = 0:plateAcquisitionList.size()-1,
            pa = plateAcquisitionList.get(i);
        end
    end
end

34https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/io/getScreens.m
To return the orphaned plates as well as the screens, you can query the second output argument of `getScreens`:

```
[screens, orphanedPlates] = getScreens(session)
```

To filter screens by owner, use the `owner` parameter/key value. A value of -1 means screens are retrieved independently of their owner:

```
% Returns all screens owned by the specified user in the context of the session group
screens = getScreens(session, 'owner', ownerId);
% Returns all screens with the input identifiers owned by the specified user
screens = getScreens(session, ids, 'owner', ownerId);
% Returns all screens owned by any user in the context of the session group
screens = getScreens(session, 'owner', -1);
```

To filter screens by group, use the `group` parameter/key value. A value of -1 means screens are retrieved independently of their group:

```
% Returns all screens owned by the session user in the specified group
screens = getScreens(session, 'group', groupId);
% Returns all screens with the input identifiers in the specified group
screens = getScreens(session, ids, 'group', groupId);
% Returns all screens owned by the session user across groups
screens = getScreens(session, 'group', -1);
```

• Plates

The screens owned by the session user in the context of the session group can be retrieved using the `getPlates` function:

```
plates = getPlates(session)
```

If the plate identifiers are known, they can be retrieved independently of their owner or group using:

```
plates = getPlates(session, ids)
```

To filter plates by owner, use the `owner` parameter/key value. A value of -1 means plates are retrieved independently of their owner:

```
% Returns all plates owned by the specified user in the context of the session group
plates = getPlates(session, 'owner', ownerId);
% Returns all plates with the input identifiers owned by the specified user
plates = getPlates(session, ids, 'owner', ownerId);
% Returns all plates owned by any user in the context of the session group
plates = getPlates(session, 'owner', -1);
```

To filter plates by group, use the `group` parameter/key value. A value of -1 means plates are retrieved independently of their group:

```
% Returns all plates owned by the session user in the specified group
plates = getPlates(session, 'group', groupId);
```

---

35https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/io/getScreens.m
36https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/io/getPlates.m
% Returns all plates with the input identifiers in the specified group
plates = getPlates(session, ids, 'group', groupId);

% Returns all plates owned by the session user across groups
plates = getPlates(session, 'group', -1);

• Wells

Given a plate identifier, the wells can be loaded using the findAllByQuery method:

wellList = session.getQueryService().findAllByQuery(
    ['select well from Well as well ...
     'left outer join fetch well.plate as pt ...
     'left outer join fetch well.wellSamples as ws ...
     'left outer join fetch ws.plateAcquisition as pa ...
     'left outer join fetch ws.image as img ...
     'left outer join fetch img.pixels as pix ...
     'left outer join fetch pix.pixelsType as pt ...
     'where well.plate.id = ', num2str(plateId)], []);

for j = 0:wellList.size()-1,
    well = wellList.get(j);
    wellsSampleList = well.copyWellSamples();
    well.getId().getValue()
    for i = 0:wellsSampleList.size()-1,
        ws = wellsSampleList.get(i);
        ws.getId().getValue()
        pa = ws.getPlateAcquisition();
    end
end

11.5.4 Raw data access

You can retrieve data, plane by plane or retrieve a stack.

• Plane

The plane of an input image at coordinates (z, c, t) can be retrieved using the getPlane function:

plane = getPlane(session, image, z, c, t);

Alternatively, the image identifier can be passed to the function:

plane = getPlane(session, imageID, z, c, t);

• Tile

The tile of an input image at coordinates (z, c, t) originated at (x, y) and of dimensions (w, h) can be retrieved using the getTile function:

tile = getTile(session, image, z, c, t, x, y, w, h);

Alternatively, the image identifier can be passed to the function:

tile = getTile(session, imageID, z, c, t, x, y, w, h);

• Stack

https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/image/getPlane.m

https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/image/getTile.m
The stack of an input image at coordinates (c, t) can be retrieved using the `getStack` function:

```matlab
stack = getStack(session, image, c, t);
```

Alternatively, the image identifier can be passed to the function:

```matlab
stack = getStack(session, imageID, c, t);
```

All the methods described above will internally initialize a raw pixels store to retrieve the pixels data and close this store at the end of the call. This is inefficient when multiple planes/tiles/stacks need to be retrieved. For each function, it is possible to initialize a pixels store and pass this store directly to the pixel retrieval function, e.g.:

```matlab
[store, pixels] = getRawPixelsStore(session, image);
for z = 0 : sizeZ - 1
   for c = 0 : sizeC - 1
      for t = 0 : sizeT - 1
         plane = getPlane(pixels, store, z, c, t);
      end
   end
end
store.close();
```

- **Hypercube**

  This is useful when you need the Pixels intensity.

```matlab
store = session.createRawPixelsStore();
store.setPixelsId(pixelsId, false);

offset = java.util.ArrayList;
offset.add(java.lang.Integer(0));
offset.add(java.lang.Integer(0));
offset.add(java.lang.Integer(0));
offset.add(java.lang.Integer(0));
offset.add(java.lang.Integer(0));

size = java.util.ArrayList;
size.add(java.lang.Integer(sizeX));
size.add(java.lang.Integer(sizeY));
size.add(java.lang.Integer(sizeZ));
size.add(java.lang.Integer(sizeC));
size.add(java.lang.Integer(sizeT));

step = java.util.ArrayList;
step.add(java.lang.Integer(1));
step.add(java.lang.Integer(1));
step.add(java.lang.Integer(1));
step.add(java.lang.Integer(1));
step.add(java.lang.Integer(1));

store.getHypercube(offset, size, step);
store.close();
```

---

39 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/image/getStack.m
11.5.5 Annotations

• Reading annotations by ID
If the identifier of the annotation of a given type is known, the annotation can be retrieved from the server using the generic `getAnnotations` function:

```matlab
tagAnnotations = getAnnotations(session, 'tag', tagIds);
```

Shortcut functions are available for the main object and annotation types, e.g. to retrieve tag annotations:

```matlab
tagAnnotations = getTagAnnotations(session, tagIds);
```

• Reading annotations linked to an object
The annotations of a given type linked to a given object can be retrieved using the generic `getObjectAnnotations` function:

```matlab
tagAnnotations = getObjectAnnotations(session, 'tag', 'image', imageIds);
```

Shortcut functions are available for the main object and annotation types, e.g. to retrieve the tag annotations linked to images:

```matlab
tagAnnotations = getImageTagAnnotations(session, imageIds);
```

Annotations can be filtered by namespace. To include only annotations with a given namespace `ns`, use the `include` parameter/key value:

```matlab
tagAnnotations = getImageTagAnnotations(session, imageIds, 'include', ns);
```

To exclude all annotations with a given namespace `ns`, use the `exclude` parameter/key value:

```matlab
tagAnnotations = getImageTagAnnotations(session, imageIds, 'exclude', ns);
```

By default, only the annotations owned by the session owner are returned. To specify the owner of the annotations, use the `owner` parameter/key value pair. For instance to return all tag annotations owned by user 5:

```matlab
tagAnnotations = getImageTagAnnotations(session, imageIds, 'owner', 5);
```

To retrieve all annotations independently of their owner, use `-1` as the owner identifier:

```matlab
tagAnnotations = getImageTagAnnotations(session, imageIds, 'owner', -1);
```

• Reading file annotations
The content of a file annotation can be downloaded to local disk using the `getFileAnnotationContent` function. If the file annotation has been retrieved from the server as `fileAnnotation`, then the content of its `OriginalFile` can be downloaded under `target_file` using:

```matlab
https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/annotations/getAnnotations.m
https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/annotations/getObjectAnnotations.m
https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/annotations/getFileAnnotationContent.m
```
getFileAnnotationContent(session, fileAnnotation, target_file);

Alternatively, if only the identifier of the file annotation faId is known:

getFileAnnotationContent(session, faId, target_file);

- Writing and linking annotations

New annotations can be created using the corresponding write*Annotation function:

```matlab
% Create a comment annotation
commentAnnotation = writeCommentAnnotation(session, 'comment');
% Create a double annotation
doubleAnnotation = writeDoubleAnnotation(session, .5);
% Create a map annotation
mapAnnotation = writeMapAnnotation(session, 'key', value);
% Create a tag annotation
tagAnnotation = writeTagAnnotation(session, 'tag name');
% Create a timestamp annotation
timestampAnnotation = writeTimestampAnnotation(session, now);
% Create an XML annotation
xmlAnnotation = writeXmlAnnotation(session, xmlString);
```

File annotations can also be created from the content of a `local_file_path`:

```matlab
fileAnnotation = writeFileAnnotation(session, local_file_path);
```

Each annotation creation function uses the context of the session group by default. To create the annotation in a different group, use the `group` key/value pair:

```matlab
commentAnnotation = writeCommentAnnotation(session, 'comment', 'group', groupId);
doubleAnnotation = writeDoubleAnnotation(session, .5, 'group', groupId);
mapAnnotation = writeMapAnnotation(session, 'key', value, 'group', groupId);
tagAnnotation = writeTagAnnotation(session, 'tag name', 'group', groupId);
timestampAnnotation = writeTimestampAnnotation(session, now, 'group', groupId);
xmlAnnotation = writeXmlAnnotation(session, xmlString, 'group', groupId);
fileAnnotation = writeFileAnnotation(session, local_file_path, 'group', groupId);
```

Existing annotations can be linked to existing objects on the server using the `linkAnnotation` function. For example, to link a tag annotation and a file annotation to the image `image_id`:

```matlab
link1 = linkAnnotation(session, tagAnnotation, 'image', image_id);
link2 = linkAnnotation(session, fileAnnotation, 'image', image_id);
```

For existing file annotations, it is possible to replace the content of the original file without having to recreate a new file annotation using the `updateFileAnnotation` function. If the file annotation has been retrieved from the server as `fileAnnotation`, then the content of its `OriginalFile` can be replaced by the content of `local_file_path` using:

```matlab
updateFileAnnotation(session, fileAnnotation, local_file_path);
```

See also:

- WriteData.m
- Examplescript showing methods to write, link and retrieve annotations

---

44https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/annotations/linkAnnotation.m
45https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/annotations/updateFileAnnotation.m
11.5.6 Writing data

• Projects/Datasets

Projects and datasets can be created in the context of the session group using the `createProject` function:

```matlab
% Create a new project in the context of the session group
newproject = createProject(session, 'project name');
% Create a new dataset in the context of the session group
newdataset = createDataset(session, 'dataset name');
```

Writing projects/datasets in a different context than the session group can be achieved by passing the group identifier using the `group` parameter:

```matlab
% Create a new project in the specified group
newproject = createProject(session, 'project name', 'group', groupId);
% Create a new dataset in the specified group
newdataset = createDataset(session, 'dataset name', 'group', groupId);
```

When creating a dataset, it is possible to link it to an existing project using either the project object or its identifier. In this case, the group context is determined by the parent project:

```matlab
% Create two new projects in different groups
project1 = createProject(session, 'project name');
project2 = createProject(session, 'project name', 'group', groupId);
% Create new datasets linked to each project
dataset1 = createDataset(session, 'dataset name', project1);
dataset2 = createDataset(session, 'dataset name', project2.getId().getValue());
```

• Screens/Plates

Screens and plates can be created in the context of the session group using the `createScreen` function:

```matlab
% Create a new screen in the context of the session group
newscreen = createScreen(session, 'screen name');
% Create a new plate in the context of the session group
newplate = createPlate(session, 'plate name');
```

Writing screens/plates in a different context than the session group can be achieved by passing the group identifier using the `group` parameter:

```matlab
% Create a new screen in the specified group
newscreen = createScreen(session, 'screen name', 'group', groupId);
% Create a new plate in the specified group
newplate = createPlate(session, 'plate name', 'group', groupId);
```

When creating a plate, it is possible to link it to an existing screen using either the screen object or its identifier. In this case, the group context is determined by the parent screen:

```matlab
% Create two new projects in different groups
screen1 = createScreen(session, 'screen name');
screen2 = createScreen(session, 'screen name', 'group', groupId);
% Create new datasets linked to each project
```

47 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/io/createProject.m
48 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/io/createDataset.m
49 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/io/createScreen.m
50 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/io/createPlate.m
plate1 = createPlate(session, 'plate name', screen1);
plate2 = createPlate(session, 'plate name', screen2.getId().getValue());

See also:
WriteData.m

11.5.7 How to use OMERO tables

• Create a table. In the following example, a table is created with 2 columns.

name = char(java.util.UUID.randomUUID());
columns = javaArray('omero.grid.Column', 2)
columns(1) = omero.grid.LongColumn('Uid', 'testLong', []);
valuesString = javaArray('java.lang.String', 1);
columns(2) = omero.grid.StringColumn('MyStringColumn', '', 64, valuesString);

$ Create a new table.
table = session.sharedResources().newTable(1, name);

$ Initialize the table
table.initialize(columns);
$ Add data to the table.
data = javaArray('omero.grid.Column', 2);
data(1) = omero.grid.LongColumn('Uid', 'test Long', []);
valuesString = javaArray('java.lang.String', 1);
valuesString(1) = java.lang.String('add');
data(2) = omero.grid.StringColumn('MyStringColumn', '', 64, valuesString);
table.addData(data);

• Read the contents of the table.

of = omero.model.OriginalFileI(file.getId(), false);
tablePrx = session.sharedResources().openTable(of);

$ Read headers
headers = tablePrx.getHeaders();
for i=1:size(headers, 1),
    headers(i).name; % name of the header
    % Do something
end

$ Depending on the size of table, you may only want to read some blocks.
cols = [0:size(headers, 1)-1]; % The number of columns you wish to read.
rows = [0:tablePrx.getNumberOfRows()-1]; % The number of rows you wish to read.
data = tablePrx.slice(cols, rows); % Read the data.
c = data.columns;
for i=1:size(c),
    column = c(i);
    % Do something
end
tablePrx.close(); % Important to close when done.

11.5.8 ROIs

To learn about the model, see the developers guide to the ROI model⁵². Note that annotations can be linked to ROI.

⁵²http://www.openmicroscopy.org/site/support/ome-model/developers/roi.html
Creating ROI

This example creates a ROI with two shapes, a rectangle and an ellipse, and attaches it to an image:

```matlab
% First create a rectangular shape.
rectangle = createRectangle(0, 0, 10, 20);
% Indicate on which plane to attach the shape
setShapeCoordinates(rectangle, 0, 0, 0);

% First create an ellipse shape.
ellipse = createEllipse(0, 0, 10, 20);
% Indicate on which plane to attach the shape
setShapeCoordinates(ellipse, 0, 0, 0);

% Create the roi.
roi = omero.model.RoiI;
% Attach the shapes to the roi, several shapes can be added.
roi.addShape(rectangle);
roi.addShape(ellipse);

% Link the roi and the image
roi.setImage(omero.model.ImageI(imageId, false));
% Save
iUpdate = session.getUpdateService();
roi = iUpdate.saveAndReturnObject(roi);
% Check that the shape has been added.
numShapes = roi.sizeOfShapes;
for ns = 1:numShapes
    shape = roi.getShape(ns-1);
end
```

See also:

ROI utility functions

OMERO.matlab functions for creating and managing Shape and ROI objects.

- Retrieving ROIs linked to an image

```matlab
service = session.getRoiService();
roiResult = service.findByImage(imageId, []);
rois = roiResult.rois;
n = rois.size;
shapeType = '';
for thisROI = 1:n
    roi = rois.get(thisROI-1);
    numShapes = roi.sizeOfShapes;
    for ns = 1:numShapes
        shape = roi.getShape(ns-1);
        if (isa(shape, 'omero.model.Rectangle'))
            rectangle = shape;
            rectangle.getX().getValue()
        elseif (isa(shape, 'omero.model.Ellipse'))
            ellipse = shape;
            ellipse.getCx().getValue()
        elseif (isa(shape, 'omero.model.Point'))
            point = shape;
            point.getX().getValue()
        elseif (isa(shape, 'omero.model.Line'))
            line = shape;
            line.getX1().getValue()
        end
    end
end
```

- Removing a shape from ROI

11.5. OMERO MATLAB language bindings
OMERO Documentation, Release 5.2.6

// Retrieve the roi linked to an image
service = session.getRoiService();
roiResult = service.findByImage(imageId, []);
n = rois.size;
for thisROI = 1:n
    roi = rois.get(thisROI - 1);
    numShapes = roi.sizeOfShapes;
    for ns = 1:numShapes
        shape = roi.getShape(ns - 1);
        % Remove the shape
        roi.removeShape(shape);
    end
    % Update the roi.
    roi = iUpdate.saveAndReturnObject(roi);
end

11.5.9 Deleting data

It is possible to delete projects, datasets, images, ROIs… and objects linked to them depending on the specified options (see Deleting in OMERO). For example, images of known identifiers can be deleted from the server using the deleteImages function:

deleteImages(session, imageIds);

See also:
deleteProjects, deleteDatasets, deleteScreens, deletePlates Utility functions to delete objects

11.5.10 Rendering images

The RenderImages.m example script shows how to initialize the rendering engine and render an image.

11.5.11 Creating Image

The CreateImage.m example script shows how to create an image in OMERO. A similar approach can be applied when uploading an image. To upload individual planes onto the server, the data must be converted into a byte (int8) array first. If the Pixels object has been created, this conversion can done using the toByteArray function.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

11.6 OMERO C++ language bindings

Using the Ice C++ language mapping from ZeroC, OMERO provides native access to your data from C++ code. CMake is used for building the C++ bindings.

Binaries are not provided, therefore it will be necessary for you to compile your own.

34https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroM/src/delete/deleteImages.m
35https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/Training/matlab/RenderImages.m
36https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/Training/matlab/CreateImage.m
37https://doc.zeroc.com/display/Ice/Hello+World+Application
38https://zeroc.com
39http://www.cmake.org/
11.6.1 Prerequisites

- The OMERO source code
- A C++ compiler
  - GCC is recommended for Linux and MacOS X
  - Visual Studio or the Platform SDK for Windows
- The ZeroC Ice libraries, headers and slice definitions
- cmake
- Google Test (optional; needed to build the unit and integration tests)

11.6.2 Restrictions

If you are restricted to a specific version of GCC or Ice, you may need to obtain or build a compatible version of Ice or GCC, respectively.

11.6.3 Preparing to build

Begin by following the instructions under Installing OMERO from source to acquire the source code. Be sure that the git branch you are using matches the version of your server!

The location of your Ice installation should be automatically detected if installed into a standard location. If this is not the case, set the location of your Ice installation using the ICE_HOME environment variable or the -DICE_HOME or -DIce_SLICE_DIR cmake options for your Ice installation (see below). Some possible locations for the 3.5.1 version of Ice follow. Note these are just examples; you need to adjust them for the Ice installation path and version in use on your system.

- Ice built from source and installed into /opt:
  export ICE_HOME=/opt/Ice-3.5.1
- Ice installed on Linux using RPM packages:
  export ICE_HOME=/usr/share/Ice-3.5.1
- MacOS X with homebrew:
  export ICE_HOME=/usr/local/Cellar/ice/3.5.1
- Windows using Visual Studio:
  set ICE_HOME=C:\Program Files (x86)\ZeroC\Ice-3.5.1

Note: If the Ice headers and libraries are not automatically discovered, these will need to be specified using appropriate cmake options (see below).

11.6.4 Building the library

For all build methods, the shared library and examples are always built by default. The unit and integration tests are built if Google test (gtest) is detected.

Building with build.py or ant

export GTEST_ROOT=/path/to/gtest
cd /path/to/openmicroscopy
./build.py build-default
./build.py build-cpp [-Dcmake.args="cmake options"]
Set any needed **cmake** options using the **ant cmake.args** property, plus any needed environment variables.

For example:

```
./build.py -Dcmake.args="'-DCMAKE_CXX_FLAGS=$CMAKE_CXX_FLAGS' \\
'-DCMAKE_EXE_LINKER_FLAGS=$CMAKE_LD_FLAGS' \\
'-DCMAKE_MODULE_LINKER_FLAGS=$CMAKE_LD_FLAGS' \\
'-DCMAKE_SHARED_LINKER_FLAGS=$CMAKE_LD_FLAGS' \\
-DCMAKE_VERBOSE_MAKEFILE:BOOL=ON"
```

This method is present for backward compatibility with the previous **scons** build system. While it is possible to customize the build somewhat, some behavior is hardcoded and so this will not be suitable for all situations and is not recommended. Parallel building is also not supported. If you wish to have full control over the C++ build or wish to use a **cmake** generator to build within an IDE, then running **cmake** directly is recommended.

### Building with **cmake** directly

On Linux, Unix or MacOS X with **make**:

```
export GTEST_ROOT=/path/to/gtest
mkdir omero-build
cd omero-build
cmake [-Dtest=(TRUE|FALSE)] [cmake options] /path/to/openmicroscopy
make
```

For example:

```
cmake "'-DCMAKE_CXX_FLAGS=$CMAKE_CXX_FLAGS' \\
"'-DCMAKE_EXE_LINKER_FLAGS=$CMAKE_LD_FLAGS' \\
"'-DCMAKE_MODULE_LINKER_FLAGS=$CMAKE_LD_FLAGS' \\
"'-DCMAKE_SHARED_LINKER_FLAGS=$CMAKE_LD_FLAGS' \\
-DCMAKE_VERBOSE_MAKEFILE:BOOL=ON /path/to/openmicroscopy
make -j8
```

Running **cmake** directly allows full use of all command-line options and provides complete flexibility for building, e.g. to enable parallel building as shown above or to use an IDE of choice with an appropriate generator.

If you would like to build the C++ tests, run the above with the **GTEST_ROOT** environment variable set.

**Note:** When **cmake** is run, it will run **./build.py build-default** in the openmicroscopy source tree to generate some of the C++ and Ice sources. If you have previously done a build by running **./build.py**, this step will be skipped. However, if you have recently switched branches without cleaning the source tree, please run **./build.py clean** in the source tree to clean up all the generated files prior to running **cmake**.

If the build fails with errors such as

```
/usr/include/Ice/ProxyHandle.h:176:13: error: ‘upCast’ was not declared in this scope, and no declarations were found by argument-dependent lookup at the point of instantiation
```

this is caused by the Ice headers being buggy, and newer versions of GCC rejecting the invalid code. To compile in this situation, add **-fpermissive** to **CXXFLAGS** to allow the invalid code to be accepted, but do note that this may also mask other problems so should not be used unless strictly needed.
11.6.5 cmake build configuration

CMake supports configuration of the build using many different environment variables and options; for a full list, see the CMake reference documentation\(^{65}\). The following environment variables are commonly needed:

**CMAKE\_INCLUDE\_PATH** Directories to be searched for include files, for example

```
/opt/Ice-3.5.1/include
```

A ; or : separator character is used to separate directories, depending on the platform. Note these are used only for feature tests, not for passing to the compiler when building, for which CMAKE\_CXX\_FLAGS is needed.

**CMAKE\_LIBRARY\_PATH** Directories to be searched for libraries, for example

```
/opt/Ice-3.5.1/lib
```

Directories are separated by ; or : as with CMAKE\_INCLUDE\_PATH. Note these are used only for feature tests and finding libraries, not for passing to the linker when building, for which CMAKE\_\*\_LINKER\_FLAGS is needed.

**CXX** C++ compiler executable. Useful with ccache\(^{66}\).

**CXXFLAGS** C++ compiler flags. Use of CMAKE\_CXX\_FLAGS is preferred.

**ICE\_HOME** The location of the Ice installation. If this is not sufficient to discover the correct binary and library directories, they may otherwise be manually specified with the options below. Likewise for the include and slice directories. This may also be set as a \texttt{cmake} cache variable (see below).

**VERBOSE** If set to \(1\), show the actual build commands rather than the pretty “Compiling XYZ...” statements.

In addition, \texttt{cmake} options may be defined directly when running \texttt{cmake}. Commonly needed options include:

-\texttt{DCMAKE\_PREFIX\_PATH} Search this location when searching for programs, headers and libraries. Use to search /usr/local or /opt/Ice, for example. More specific search locations may be specified using \texttt{DCMAKE\_INCLUDE\_PATH}, \texttt{DCMAKE\_LIBRARY\_PATH} and \texttt{DCMAKE\_PROGRAM\_PATH} separately, if required.

-\texttt{DCMAKE\_INCLUDE\_PATH} Search this location when searching for headers. Use to include /usr/local/include or /opt/Ice/include, for example.

-\texttt{DCMAKE\_LIBRARY\_PATH} Search this location when searching for libraries. Use to include /usr/local/lib or /opt/Ice/lib, for example.

-\texttt{DCMAKE\_PROGRAM\_PATH} Search this location when searching for programs. Use to include /usr/local/bin or /opt/Ice/bin, for example.

-\texttt{DCMAKE\_CXX\_FLAGS} C++ compiler flags. Use to set any additional linker flags desired.

-\texttt{DCMAKE\_EXE\_LINKER\_FLAGS} Executable linker flags. Use to set any additional linker flags desired.

-\texttt{DCMAKE\_MODULE\_LINKER\_FLAGS} Loadable module linker flags. Use to set any additional linker flags desired.

-\texttt{DCMAKE\_SHARED\_LINKER\_FLAGS} Shared library linker flags. Use to set any additional linker flags desired.

-\texttt{DCMAKE\_VERBOSE\_MAKEFILE} Default to printing all commands executed by make. This may be overridden with the make \texttt{VERBOSE} variable.

-\texttt{DICE\_HOME} The location of the Ice installation. If this is not sufficient to discover the correct binary and library directories, they may otherwise be manually specified with the options below. Likewise for the include and slice directories.

-\texttt{DICE\_SLICE2XXX\_EXECUTABLE} Specific location of individual Ice slice2xxx programs, e.g. Ice\_SLICE2CPP\_EXECUTABLE for slice2cpp or Ice\_SLICE2JAVA\_EXECUTABLE for slice2java. These are typically found in \$\{ICE\_HOME\}/bin or on the default PATH. These will not normally require setting.

-\texttt{DICE\_INCLUDE\_DIR} Location of Ice headers. This is typically \$\{ICE\_HOME\}/include or on the default include search path. This will not normally require setting.

-\texttt{DICE\_SLICE\_DIR} Location of Ice slice interface definitions. This is typically \$\{ICE\_HOME\}/slice. Use for installations where \texttt{DICE\_HOME} does not contain slice or situations where you wish to build without setting \texttt{DICE\_HOME}. Note that when building using \texttt{build.py}, rather than building directly with \texttt{cmake}, the SLICEPATH environment variable should be used instead (the \texttt{ant} build can’t use the \texttt{cmake} variables since it only runs \texttt{cmake} after a full build of the Java server).

\(^{65}\)http://www.cmake.org/cmake/help/documentation.html

\(^{66}\)http://ccache.samba.org/
-DIce_<C>_LIBRARIES Specific libraries for Ice component <C>, where <C> is the uppercased name of the Ice component. e.g. ICE for the Ice component, ICEUTIL for the IceUtil component or GLACIER2 for the Glacier2 component. These libraries are typically found in $ICE_HOME/lib or on the default library search path. These will not normally require setting.

-DIce_DEBUG Set to ON to print detailed diagnostics about the detected Ice installation. Use if there are any problems finding Ice.

cmake offers many additional options. Please refer to the documentation for further details, in particular to the variables which change the behavior of the build.

### 11.6.6 Visual Studio configuration

**Warning:** OMERO.cpp will not currently build on Windows due to exceeding DLL symbol limits on this platform, leading to a failure when linking the DLL. It is hoped that this platform limitation can be worked around in a future OMERO release.

Cmake has full support for Visual Studio. Use the cmake -G option to set the generator for your Visual Studio version, with a Win64 suffix for an x64 build. The correct Ice programs and libraries for your Ice installation should be automatically discovered.

```cmake
cmake -G "Visual Studio 10 Win64" [cmake options] /path/to/openmicroscopy
```

This is for a 64-bit Visual Studio 2010 build. Modify appropriately for other versions and compilers. Running

```cmake --help```

will list the available generators for your platform (without the Win64 suffix).

Once cmake has finished running, the generated project and solution files may be then opened in Visual Studio, or built directly using the msbuild command-line tool (make sure that the Visual Studio command prompt matches the generator chosen) or by running:

```cmake --build .```

As for the Unix build, above, it is also possible to build on Windows using build.py or ant, providing that you configure the generator appropriately using the correct cmake options. However, this will not work for all generators reliably, and the Windows shell quoting makes passing nested quotes to ant quite tricky, so running cmake by hand is recommended.

**Note:** It may be necessary to specify /Zm1000 as an additional compiler setting.

### 11.6.7 Installing the library

If using make, run:

```make [DESTDIR=/path/to/staging/directory] install```

If using another build system, please invoke the equivalent install target for that system.

### 11.6.8 Using the library

To use *OMERO C++ language bindings* it is necessary to point your compiler and linker at the mentioned directories above. A simple GNU make Makefile might look like this:

67http://www.cmake.org/cmake/help/v3.0/
68http://www.cmake.org/cmake/help/v3.0/manual/cmake-variables.7.html#variables-that-change-behavior
# MAKEFILE:

# Where the OMERO distribution was installed
OMERO_DIST?= /opt/omero

# Where the Ice lib/ and include/ directories are to be found
ICE_HOME?= /usr/share/Ice

INCLUDES=-I$(OMERO_DIST)/include -I$(ICE_HOME)/include

LIBS = -L$(OMERO_DIST)/lib -L$(ICE_HOME)/lib -L$(ICE_HOME)/lib64 \ 
       -lIce -lIceUtil -lGlacier2 -lomero-client

LIBPATH = $(LD_LIBRARY_PATH):$(ICE_HOME)/lib:$($(ICE_HOME)/lib64):$(OMERO_DIST)/lib

.PHONY: clean run

yourcode.o: yourcode.cpp
  $(CXX) $(CXXFLAGS) -c -o $@ $< $(INCLUDES)

yourcode: yourcode.o
  $(CXX) -o $@ $^ $(LIBS)

run: yourcode
  LD_LIBRARY_PATH="$(LIBPATH)" ./yourcode --Ice.Config=./etc/ice.config

clean:
  rm -f yourcode *.o *~ core

11.6.9 A trivial example: yourcode.cpp

A simple example might look something like the following:

//
// yourcode.cpp:
//

// Domain
#include <omero/client.h>
#include <omero/api/IAdmin.h>

// Std
#include <iostream>
#include <cassert>
#include <vector>
#include <time.h>
#include <map>

using namespace std;

/*
 * Pass "--Ice.Config=your_config_file" to the executable, or
 * set the ICE_CONFIG environment variable.
*/
int main(int argc, char* argv[])
{
  omero::client_ptr omero = new omero::client(argc, argv);
  omero::api::ServiceFactoryPrx sf = omero->createSession();
  sf->closeOnDestroy();

  // IAdmin is responsible for all user/group creation, password changing, etc.
omero::api::IAdminPrx admin = sf->getAdminService();

// Who you are logged in as.
cout << admin->getEventContext()->userName << endl;

// These two services are used for database access
omero::api::IQueryPrx query = sf->getQueryService();
omero::api::IUpdatePrx update = sf->getUpdateService();

return 0;
}

This code does not do much. It creates a server session, loads a few services, and prints the user’s name. For serious examples, see Working with OMERO.

### 11.6.10 Compiling and running your code

To compile and run your code, download the two files above (Makefile and yourcode.cpp) and then in a shell:

```
make OMERO_DIST=dist yourcode
LD_LIBRARY_PATH=dist/lib ./yourcode --Ice.Config=dist/etc/ice.config
```

where you have edited dist/etc/ice.config to contain the values:

```plaintext
omero.host=localhost
omero.user=your_name
omero.pass=your_password
```

Alternatively, you can pass these on the command-line:

```
LD_LIBRARY_PATH=dist/lib ./yourcode omero.host=localhost --omerost.user=foo --omerost.pass=bar
```

*Note:* This example explains how to run on Linux only. For doing the same on MacOS X, change all instances of `LD_LIBRARY_PATH` to `DYLD_LIBRARY_PATH`.

### 11.6.11 Further information

For the details behind writing, configuring, and executing a client, please see Working with OMERO.

See also:

Ice\(^{69}\), OMERO.grid, OMERO Application Programming Interface, Build System, #1596\(^{70}\) which added 64-bit support

---

\(^{69}\)https://zeroc.com

\(^{70}\)https://trac.openmicroscopy.org/ome/ticket/1596
12.1 Local analysis

If you are interested in running your analysis locally and storing the results to the server, then your first step is to become familiar with the developer documentation.

- The Working with OMERO guide provides numerous examples in each language with explanations and tries to be a starting point for anyone who wants to write code which talks to the OMERO server.
- Most of the OMERO Application Programming Interface is covered by the Javadocs.
- Each of the languages has extra information on its own page:
  - OMERO C++ language bindings
  - OMERO Java language bindings
  - OMERO MATLAB language bindings
  - OMERO Python language bindings

Once you have your local analysis working, you can push it onto the server for background processing using the OMERO scripting service.

12.2 Storing external data in OMERO

There are several options for storing external or schema-less data in OMERO, including StructuredAnnotations for small quantities of data, or extending the OME model, but this risks interoperability issues. (See Extending Omero).

For larger volumes of data, or data which needs to be queried, OMERO.tables provides a unified solution for the storage of columnar data from various sources, such as automated analysis results or script-based processing, and makes them available within OMERO.

12.2.1 Third-party analysis and OMERO.tables

Support has been added for some third-party analysis data, which gets converted in OMERO into a common format. These formats include:

- MIAS data, measurements, and overlays
- InCell data and measurements
- Flex data with Acapella results (screencast\(^2\)). In the Flex case, additional configuration may be necessary for accessing both the raw data and the analysis results. Watch the configuration screencast\(^3\) for more information.

---

The analysis results which are parsed out of the formats listed above are converted to HDF by the OMERO.tables API. This facility can then be used by clients to visualize the parsed measurements, and in the case of regions of interest, see their location overlayed on the associated image:

12.2.2 Other high-content screening (HCS) data

In addition to the Flex, Mias, and InCell 100 file formats, BD Pathway, Olympus ScanR, and native OME-XML/TIFF files can all be imported as HCS data, though without support for any external analysis data which may be attached. If you are interested in having other analysis formats supported, contact either the open source community or Glencoe Software, Inc.⁴ depending on your needs.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

12.3 OMERO.tables

The OMERO.tables API unifies the storage of columnar data from various sources, such as automated analysis results or script-based processing, and makes them available within OMERO.

Large and small volumes of tabular data can be stored via named columns, and retrieved in bulk or via paging. A limited query language provides basic filtering and selecting.

For installation instructions, see OMERO.tables

12.3.1 The interface

The slice definition file⁵ for the OMERO.tables API primarily defines two service interfaces and a type hierarchy.

**class omero.grid.Table** The central service for dealing with tabular data, described below.

⁴http://www.glencoesoftware.com/
⁵https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/resources/omero/Tables.ice
class omero.grid.Tables
An internal service used for managing table services, and can be ignored for almost all purposes.

class omero.grid.Column
The base class for column types which permit returning arrays of columnar values (Ice\(^6\) doesn’t provide an Any type, so it is necessary to group values of the same type). All columns in a table must have the same number of rows.

Note: Attribute names (including column names) beginning with `__` (double underscore) are reserved for internal use. This restriction was introduced in OMERO 5.1, Tables created by older versions should continue to work.

### Single value columns

These columns store a single value in each row.

```python
class omero.grid.FileColumn(name, description[, values])
class omero.grid.ImageColumn(name, description[, values])
class omero.grid.RoiColumn(name, description[, values])
class omero.grid.WellColumn(name, description[, values])
class omero.grid.PlateColumn(name, description[, values])
```

Id-based (long) columns which reference `omero.model.File`, Image, Roi, Well and Plate instances respectively.

```python
class omero.grid.BoolColumn(name, description[, values])
A value column with bool (non-null) values.
class omero.grid.LongColumn(name, description[, values])
A value column with long (non-null, 64-bit) values.
class omero.grid.DoubleColumn(name, description[, values])
A value column with double (non-null, 64-bit) values.
```

**Parameters**

- `name (string)` – The name of the column, each column in a table must have a unique name.
- `description (string)` – The column description, may be empty.
- `values ([])` – A list of values (one value per row) used to initialize a column (optional).

**values**
A class member holding the list of values stored in the column.

```python
class omero.grid.StringColumn(name, description, size[, values])
A value column which holds strings
```

**Parameters**

- `name (string)` – The column name.
- `description (string)` – The column description.
- `size (long)` – The maximum string length that can be stored in this column, >= 1
- `values (string[])` – A list of strings (optional).

### Array value columns

These columns store an array in each row.

```python
class omero.grid.FloatArrayColumn(name, description, size[, values])
A value column with fixed-width arrays of float (32 bit) values.
class omero.grid.DoubleArrayColumn(name, description, size[, values])
A value column with fixed-width arrays of double (64 bit) values.
class omero.grid.LongArrayColumn(name, description, size[, values])
A value column with fixed-width arrays of long (64 bit) values.
```

---

6https://zeroc.com
Parameters

- **name** *(string)* – The column name.
- **description** *(string)* – The column description.
- **size** *(long)* – The width of the array, \( \geq 1 \)
- **values** *(list)* – A list of arrays, each of length `size` (optional).

**Warning:** The OMERO.tables service currently does limited validation of string and array lengths. When adding or modifying data it is essential that the `size` parameter of a column matches that of the underlying table.

**Warning:** Array value columns should be considered experimental for now.

Main methods

```python
class omero.grid.Data
    Holds the data retrieved from a table, also used to update a table.
    
    lastModification
        The timestamp of the last update to the table.
    
    rowNumbers
        The row indices of the values retrieved from the table.
    
    columns
        A list of columns

class omero.grid.Table
    The main interface to the Tables service.
    
    getHeaders()
        Returns An empty list of columns describing the table. Fill in the values of these columns to add a new row to the table.
    
    getNumberOfRows()
        Returns The number of rows in the table.
    
    readCoordinates(rowNumbers)
        Read a set of entire rows in the table.

        Parameters rowNumbers *(long[])* – A list of row indices to be retrieved from the table.

        Returns The requested rows as a Data object.
    
    read(colNumbers, start, stop)
        Read a subset of columns and consecutive rows from a table.

        Parameters

            - colNumber *(long[])* – A list of column indices to be retrieved from the table (may be non-consecutive).
            - start *(long)* – The index of the first row to retrieve.
            - stop *(long)* – The index of the last+1 row to retrieve (uses similar semantics to range()).

        Returns The requested columns and rows as a Data object.

        Note: `start=0, stop=0` currently returns the first row instead of empty as would be expected using the normal Python range semantics. This may change in future.
    
    slice(colNumbers, rowNumbers)
        Read a subset of columns and rows (may be non-consecutive) from a table.

        Parameters
```

12.3. OMERO.tables
• **colNumbers** (*long[ ]*) – A list of column indices to be retrieved. The results will be returned in the same order as these indices.

• **rowNumbers** (*long[ ]*) – A list of row indices to be retrieved. The results will be returned in the same order as these indices.

**Returns** The requested columns and rows as a *Data* object.

**getWhereList** (*condition, variables, start, stop, step*)

Run a query on a table, see *Query language*.

**Parameters**

• **condition** (*string*) – The query string

• **variables** – A mapping of strings and variable values to be substituted into *condition*. This can often be left empty.

• **start** (*long*) – The index of the first row to consider.

• **stop** (*long*) – The index of the last+1 row to consider.

• **step** (*long*) – The stepping interval between the *start* and *stop* rows to consider, using the same semantics as *range()* . Set to 0 to disable stepping.

**Returns** A list of row indices matching the condition which can be passed as the first parameter of *readCoordinates()* or *read()*.

**Note:** *variables* seems to add unnecessary complexity, should it be removed?

**initialize** (*columns*)

Initialize a new table. Any column values are ignored, use *addData()* to add these values.

**Parameters**

• **columns** (*Column[ ]*) – A list of columns whose names and types are used to setup the table.

**addData** (*columns*)

Append one or more full rows to the table.

**Parameters**

• **columns** (*Column[ ]*) – A list of columns, such as those returned by *getHeaders()* , whose values are the rows to be added to the table.

**update** (*data*)

Modify one or more columns and/or rows in a table.

**Parameters**

• **data** (*Data*) – A *Data* object previously obtained using *read()* or *readCoordinates()* with column values to be updated.

**setMetadata** (*key, value*)

Store additional properties associated with a Table.

**Parameters**

• **key** (*string*) – A key name.

• **value** (*string/int/float/long*) – The value of the property.

**setAllMetadata** (*keyvalues*)

Store multiple additional properties associated with a Table. See *setMetadata()*.

**Parameters**

• **keyvalues** (*dict*) – A dictionary of key-value pairs.

**getMetadata** (*key*)

Get the value of a property.

**Parameters**

• **key** (*string*) – The property name.

**Returns** A property.

**getAllMetadata** ()

Get all additional properties. See *getMetadata()*.

**Returns** All key-value properties.
You many find the Python and Java annotated code samples helpful, in addition to the examples and documentation on the API\(^7\). These are only an introduction to using OMERO.tables and do not show its full potential, see Going forward for some inspiration.

### 12.3.2 Examples

- **Hello World**: examples/OmeroTables/first.py\(^8\)
- **Creating a Measurement Table**: examples/OmeroTables/MeasurementTable.java\(^9\)
- **Querying a Table**: examples/OmeroTables/FindMeasurements.java\(^10\)

### 12.3.3 The implementation

Currently, each table is backed by a single HDF table. Since PyTables (and HDF in the general case) do not support concurrent access, OMERO.tables provides a global locking mechanism which permits multiple views of the same data. Each OMERO.tables file (registered as an OriginalFile in the database), is composed of a single HDF table with any number of certain limited column types.

### 12.3.4 Query language

The query language mentioned above is currently the PyTables condition syntax\(^11\). Columns are referenced by name. The following operators are supported:

- **Logical operators**: `&`, `|`, `~`
- **Comparison operators**: `<`, `<=`, `==`, `!=`, `>=`, `>`
- **Unary arithmetic operators**: `-`
- **Binary arithmetic operators**: `+`, `-`, `*`, `/`, `**`, `%`

and the following functions:

- **where**(bool, number1, number2): `number` — `number1` if the bool condition is true, `number2` otherwise.
- `{sin,cos,tan}`(float|complex): `float|complex` — trigonometric sine, cosine or tangent.
- `{arcsin,arccos,arctan}`(float|complex): `float|complex` — trigonometric inverse sine, cosine or tangent.
- `{arctan2}`(float1, float2): `float` — trigonometric inverse tangent of float1/float2.
- `{sinh,cosh,tanh}`(float|complex): `float|complex` — hyperbolic sine, cosine or tangent.
- `{arcsinh,arccosh,arctanh}`(float|complex): `float|complex` — hyperbolic inverse sine, cosine or tangent.
- `{log,log10,log1p}`(float|complex): `float|complex` — natural, base-10 and log(1+x) logarithms.
- `{exp,expm1}`(float|complex): `float|complex` — exponential and exponential minus one.
- `sqrt`(float|complex): `float|complex` — square root.
- `{real,imag}`(complex): `float` — real or imaginary part of complex.
- `complex`(float, float): `complex` — complex from real and imaginary parts.

For example, if `id` is the name of a `LongColumn`

```python
table.getWhereList(condition='(id>x)', variables={'x':omero.rtypes.rint(5)},
                          start=2, stop=10, step=3)
```

will extract a subset of rows (2, 5, 8) as indicated by `start`, `stop` and `step`, substitute 5 in place of `x` in the condition, and evaluate condition so as to return the indices of rows where column id is greater than 5.

---

\(^7\)http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/grid/Table.html
\(^8\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroTables/first.py
\(^9\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroTables/MeasurementTable.java
\(^10\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroTables/FindMeasurements.java
\(^11\)http://pytables.github.com/usersguide/condition_syntax.html
12.3.5 Going forward

The Tables API itself provides little more than a remotely accessible store, think of it as a server for Excel-like spreadsheets. We are currently looking into the facilities that can be built on top of it, and are very open to suggestions. For example, the IRoi interface\textsuperscript{12} has been extended to filter ROIs by a given measurement. This allows seeing only those results from a particular analysis run. The following example shows how to set up such a measurement and retrieve its results:

iroi.py\textsuperscript{13}

For an example of production code that parses out such measurements, see populate_roi.py\textsuperscript{14}.

The IRoi interface has been integrated into OMERO.insight, allowing for the visualization and export of OMERO.tables:

![Figure 12.1: Choice between multiple measurements](image)

We are also looking into a NoSQL-style storage mechanism for OMERO, either as an alternative back-end to OMERO.tables or as an additional key-value type store. Any suggestions or ideas would be very welcome.

See also:

- PyTables\textsuperscript{15} Software on which OMERO.tables is built.
- Condition Syntax\textsuperscript{16} The PyTables condition syntax.
- Tables.ice\textsuperscript{17} The API definition for OMERO.tables
- The Tables test suite\textsuperscript{18} The test suite for OMERO.tables
- OMERO.tables Installation requirements for install OMERO.tables

\textsuperscript{12}http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/IRoi.html

\textsuperscript{13}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroTables/iroi.py

\textsuperscript{14}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroPy/src/omero/util/populate_roi.py
13.1 Introduction to OMERO.scripts

OMERO.scripts are the OME version of plugins, allowing you to extend the functionality of your OMERO installation.

The OMERO scripting service allows scripts to be uploaded to the server so that image processing and analysis, and other functionality, can be carried out there rather than on your local machine. Scripts are generally written in Python, but other languages are also supported, like Jython. MATLAB scripts are supported natively as well as via a Python wrapper as described in MATLAB and Python.

Scripts can be run from the OMERO clients, using a UI generated from the script and the results should also be handled where relevant e.g. allowing users to view OMERO Images or Datasets created by the script, or download files or images.

Figure 13.1: Scripts menu in OMERO.insight

13.1.1 Finding scripts

Core scripts\(^1\) are bundled with every OMERO.server release and automatically available to all users. You can find additional

\(^1\)https://github.com/ome/scripts
13.1.2 Installing and running scripts

The easiest way to make use of scripts is for someone with admin rights to upload them to the OMERO.server as described in the OMERO.scripts user guide. Once a script has been added under the lib/scripts directory, you can run them from the OMERO clients or the command line.

13.1.3 Writing scripts

OMERO.scripts user guide describes the workflows for developing and running your own scripts. You should use the Guidelines for writing OMERO.scripts to ensure your script interacts with the OMERO clients in a usable way.

scripts via the new script sharing\(^2\) page.

\(^2\)http://www.openmicroscopy.org/site/community/scripts
If you are a biologist with no previous coding experience, you may find the Python for Biologists\(^3\) free online course helpful.

### 13.1.4 Managing scripts

To keep your scripts up to date, we recommend you use a Github repository to manage your scripts. If you are not familiar with using git\(^4\), you can use the GitHub app for your OS\(^5\) (available for Mac and Windows but not Linux). The basic workflow is:

- fork our omero-user-script\(^6\) repository
- clone it in your lib/scripts directory

```
cd lib/scripts;
git clone git@github.com:YOURGITUSERNAME/omero-user-scripts.git YOUR_SCRIPTS
```

- save the scripts you want to use into the appropriate sub-directory in your cloned location lib/scripts/YOUR_SCRIPTS

Your new scripts will then show up in the script menu in the clients, alongside the core ‘omero’ scripts which are shipped with each release. This means you should try to pick unique names to avoid future clashes e.g. Custom_Scripts/Search_Scripts/original_metadata_search.py:

![Figure 13.4: Custom scripts in OMERO.web menu](image)

The OME developers use Github to co-ordinate all our development work so joining the network will help you access help and support, and see what other people are doing with scripts. Cloning our repository also means you have an example script to get you started with developing your own.

### 13.1.5 Contributing back to the community

If you have modified one of the core scripts or developed your own that you would like to contribute back to the community, please get in touch. We can either add your repository to the list on the script sharing\(^7\) page so people can find it, or if the script is likely to have wide appeal, we can look into adding it to the core scripts that are distributed with an OMERO release.

See also:

- OMERO.scripts user guide,
- Guidelines for writing OMERO.scripts,
- OMERO.scripts advanced topics
- MATLAB and Python

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

---

\(^3\)http://pythonforbiologists.com/index.php/introduction-to-python-for-biologists/

\(^4\)http://www.openmicroscopy.org/site/support/contributing/using-git.html

\(^5\)http://help.github.com/articles/set-up-git

\(^6\)https://github.com/ome/omero-user-scripts

\(^7\)http://www.openmicroscopy.org/site/community/scripts
13.2 OMERO.scripts user guide

**OMERO.blitz** provides a service to run scripts on the server. The scripts are then passed on to a grid of processors called **OMERO.grid** that executes the script and returns the result to the server which in turn passes the result onto the caller. All scripts are of the form:

```python
# import the omero package and the omero.scripts package.
import omero, omero.scripts as script

This method creates the client script object, with name SCRIPTNAME and SCRIPTDESCRIPTION. The script then supplies a number of variables that are both inputs and outputs to the script. The types allowed are defined in the script package, with the qualifier after the variable of in, out or inout depending on whether the variable if for input, output or input and output.

```python
client = script.client("SCRIPTNAME", "SCRIPTDESCRIPTION",
    script.TYPE("VARIABLENAME").[in()|out()|inout()], ...)

# create a session on the server.
client.createSession()

# All variables are stored in a map accessed by getInput and setOutput via the client object.
VARIABLENAME = client.getInput("VARIABLENAME");
client.setOutput("VARIABLENAME", value);
```

This is a guide to getting started with the scripting service, without going into the ‘behind the scenes’ details. More technical details can be found on the [OMERO.scripts advanced topics page](https://github.com/ome/scripts). In addition to this guide, you may find the following pages useful for more information on using the OME Python API: Working with OME, OME Python language bindings.

### 13.2.1 Sample scripts

Below are two sample scripts. You can find the core scripts that are distributed with the OME server under the scripts repository[^8] or download them from OME.insight (from the bottom-left of any run-script dialog), or use the script sharing[^9] page to find scripts written by other users.

#### Ping script

This script echoes the input parameters as outputs.

```python
import omero, omero.scripts as script
client = script.client("ping.py", "simple ping script",
    script.Long("a"), script.String("b"))
client.createSession()

keys = client.getInputKeys()
print "Keys found:"
print keys
for key in keys:
    client.setOutput(key, client.getInput(key))
```

#### Accessing an Image and Channels on the server

This example shows usage of the Python Blitz Gateway to access an Image, using its ID. We then list the Channel names and the script returns them as outputs.

[^8]: https://github.com/ome/scripts
[^9]: http://www.openmicroscopy.org/site/community/scripts
import omero, omero.scripts as scripts
from omero.gateway import BlitzGateway
from omero.rtypes import wrap

# Define the script name & description, and a single 'required' parameter
client = scripts.client("Get_Channels.py", "Get channel names for an image",
    scripts.Long("imageId", optional=False))

# get the Image Id from the parameters.
imageId = client.getInput("imageId", unwrap=True)  # unwrap the rtype

# Use the Python Blitz Gateway for convenience
conn = BlitzGateway(client_obj=client)

# get the Image, print its name
image = conn.getObject("Image", imageId)
print image.getName()

# Print each channel 'label' (Name or Excitation wavelength)
for i, ch in enumerate(image.getChannels()):
    print ch.getLabel()

# Return as output. Key is string, value is rtype
client.setOutput("Channel%02d" % i, wrap(str(ch.getLabel())))

# Cleanup
client.closeSession()

---

Dynamic arguments

In general, script parameters are processed on server startup and cached for later use. If you have a script which should recalculate its arguments on each invocation, use the NSDYNAMIC namespace:

---

# A list of datasets will be dynamically generated and used to populate the # script parameters every time the script is called

import omero
import omero.gateway
from omero import scripts
from omero.rtypes import rstring

def get_params():
    try:
        client = omero.client()
        client.createSession()
        conn = omero.gateway.BlitzGateway(client_obj=client)
        conn.SERVICE_OPTS.setOmeroGroup(-1)
        objparams = [rstring('Dataset:%d %s' % (d.id, d.getName()))
                     for d in conn.getObjects('Dataset')]
        if not objparams:
            return [rstring('<No objects found>')]  
        return objparams
    except Exception as e:
        return ['Exception: %s' % e]
finally:
    client.closeSession()

def runScript():
    ""
    The main entry point of the script
    """
**13.2.2 Script writing as ‘Admin’**

The basic steps in a script-writing workflow are:

- Write a script using your favorite text editor, save locally
- Use command line (or OMERO.insight) to upload script to server
- Use command line (or OMERO.insight or web clients) to run script on the server (results will be displayed)
- Edit script and replace copy on server and run again, etc.

Working with scripts is far more straightforward if you have admin access to your OMERO.server installation - this is the preferred workflow. It is possible to work with scripts as a regular user (see **OMERO.scripts advanced topics**) but the software you would be required to install means it is easier to install a server on your local machine so you have admin rights.

It is assumed that scripts written by a server admin are “trusted” to run on the server without causing any disruption or security risks. Once uploaded these scripts are available to all regular users of the server alongside the official scripts included in each OMERO release.

**Download / Edit script**

The easiest way to get started is to take an existing script and edit it for your needs. An example created for the purpose of this tutorial can be found at **Edit_Descriptions.py**[10]. You should organize your scripts on your local machine in a way that makes sense to users, since your local file paths will be mimicked on the server and used to organize script menus in OMERO.insight (see screen-shot above).

```python
... objparams = get_params()

client = scripts.client(
    'Example Dynamic Test', 'Example script using dynamic parameters',
    scripts.String(
        'Dataset', optional=False, grouping='1',
        description='Select a dataset', values=objparams),
    namespaces=[omero.constants.namespaces.NSDYNAMIC],
)

try:
    scriptParams = client.getInputs(unwrap=True)
    message = 'Params: $s
% scriptParams
    print message
    client.setOutput('Message', rstring(str(message)))

finally:
    client.closeSession()

if __name__ == '__main__':
    runScript()

Example_Dynamic_Script.py
```

The action of this script (editing Image descriptions) is trivial but it demonstrates a number of features that you may find useful, including conventions for inputs and outputs to improve interaction with OMERO.insight (as discussed on the Guidelines for writing OMERO.scripts).

The script is well documented and should get you started. A few points to note:

Since the OMERO 4.3 release, if you are using the ‘Blitz Gateway’, you can get a connection wrapper like this:

```python
from omero.gateway import BlitzGateway

conn = BlitzGateway(client_obj=client)
# now you can do E.g. conn.getObject("Image", imageId) etc.
```

Alternatively, if you are working directly with the OMERO services, you can get a service factory like this:

```python
session = client.getSession()
# now you can do E.g. session.getQueryService() etc.
```

More example scripts

Several official scripts are included in the release of OMERO and can be found under the lib/scripts/omero/ directory of the server package. Any script can also be download from the OMERO.insight client (bottom-left of the run-script dialog).

**Warning:** If you wish to edit the official scripts that are part of the OMERO release, you should be prepared to apply the same changes to future releases of these scripts from OMERO. If you think that your changes should be included as part of future released scripts, please let us know.

Upload script

You can use the command line, OMERO.insight or the server file system to upload scripts. The script command line tool is discussed in more detail below.

You may find it useful to add the OMERO.server/bin/ folder to your PATH so you can call bin/omero commands when working in the scripts folder. E.g:

```bash
export PATH=$PATH:/Users/will/Desktop/OMERO.server-5.2.6/bin/
```

Upload the script we saved earlier, specifying it as ‘official’ (trusted to run on the server processor). You will need to log in the first time you use the omero script command. The new script ID will be printed out:

```bash
$ cd Desktop/scripts/
$ omero script upload demo_tutorial/Edit_Descriptions.py --official
Previously logged in to localhost:4064 as root
Server: [localhost]           # hit 'enter' to accept default login details
Username: [root]
Password:
Created session 09fcf689-cc85-409d-91ac-f9865dbfd650 (root@localhost:4064). Idle timeout: 10.0 min. Current group: system
Uploaded official script as original file #301
```

You can add, remove and edit scripts directly in the OMERO_HOME/lib/scripts/omero/ folder. Any changes made here will be detected by OMERO. Official scripts are uniquely identified on the OMERO server by their ‘path’ and ‘name’.

Any folders in the script path are created on the server under /lib/scripts/ E.g. the above example will be stored at /lib/scripts/examples/Edit_Descriptions.py

The ID of the script is printed after upload and can also be determined by listing scripts (see below).
Run script

You can run the script from OMERO.insight by browsing the scripts (see screen-shot above). A UI will be generated from the chosen script and the currently selected images or datasets will be populated if the script supports this (see Guidelines for writing OMERO.scripts).

Or launch the script from the command line, specifying the script ID. You will be asked to provide input for any non-optional parameters that do not have default values specified. Any stdout and stderr will be displayed as well as any outputs that the script has returned.

```
wm:examples will$ omero script launch 301 # script ID
Using session 1202acc0-4424-4fa2-84fe-7c9e069d3563 (root@localhost:4064). Idle timeout: 10.0 min. Current user: wjm
Enter value for "IDs": 1201
Job 1464 ready
Waiting....
Callback received: FINISHED

*** start stdout ***
* {'IDs': [1201L], 'Data_Type': 'Dataset'}
* Processing Images from Dataset: LSM - .mdb
* Editing images with this description:
* No description specified
* Editing image ID: 15651 Name: sample files.mdb [XY-ch-02]
* Editing image ID: 15652 Name: sample files.mdb [XY-ch-03]
* Editing image ID: 15653 Name: sample files.mdb [XY-ch]
* Editing image ID: 15654 Name: sample files.mdb [XYT]
* Editing image ID: 15655 Name: sample files.mdb [XYZ-ch-20x]
* Editing image ID: 15656 Name: sample files.mdb [XYZ-ch-zoom]
* Editing image ID: 15658 Name: sample files.mdb [XYZ-ch0]
* Editing image ID: 15657 Name: sample files.mdb [XYZ-ch]

*** end stdout ***

*** out parameters ***
* Message=8 Images edited
*** done ***
```

Parameter values can also be specified in the command.

```
# simply specify the required parameters that don't have defaults
$ omero script launch 301 IDs=1201

# can also specify additional parameters
$ omero script launch 301 Data_Type='Image' IDs=15652,15653 New_Description="Adding description from script to Image"
```

Edit and replace

Edit the script and upload it to replace the previous copy, specifying the ID of the file to replace.

```
$ omero script replace 301 examples/Edit_Descriptions.py
```

Finally, you can upload and run your scripts from OMERO.insight.

Other script commands

Start by printing help for the script command:
$ omero script -h
usage: /Users/will/Documents/workspace/Omero/dist/bin/omero script
       [-h] <subcommand> ...

Support for launching, uploading and otherwise managing OMERO.scripts

Optional Arguments:
In addition to any higher level options
-h, --help show this help message and exit

Subcommands:
Use /Users/will/Documents/workspace/Omero/dist/bin/omero script <subcommand> -h for more information.

<subcommand>
demo Runs a short demo of the scripting system
list List files for user or group
cat Prints a script to standard out
edit Opens a script in $EDITOR and saves it back to the server
params Print the parameters for a given script
launch Launch a script with parameters
disable Makes script non-executable by setting the mimetype
enable Makes a script executable (sets mimetype to text/x-python)
jobs List current jobs for user or group
serve Start a usermode processor for scripts
upload Upload a script
replace Replace an existing script with a new value
run Run a script with the OMERO libraries loaded and current login

To list scripts on the server:

$ omero script list
Using session 09f689-cc85-409d-91ac-f9865dbfd650 (root@localhost:4064). Idle timeout: 10.0 min. Current group: system
id | Official scripts
-----+---------------------------------------------
201 | /omero/analysis_scripts/flim-omero.py
1 | /omero/analysis_scripts/FLIM.py
202 | /omero/export_scripts/Batch_Image_Export.py
203 | /omero/export_scripts/Make_Movie.py
204 | /omero/figure_scripts/Movie_Figure.py
205 | /omero/figure_scripts/Movie_ROI_Figure.py
206 | /omero/figure_scripts/ROI_Split_Figure.py
207 | /omero/figure_scripts/Split_View_Figure.py
208 | /omero/figure_scripts/Thumbnail_Figure.py
8 | /omero/import_scripts/Populate_ROI.py
9 | /omero/setup_scripts/FLIM_initialise.py
209 | /omero/util_scripts/Channel_Offsets.py
210 | /omero/util_scripts/Combine_Images.py
211 | /omero/util_scripts/Images_From_ROIs.py
(14 rows)

If you want to know the parameters for a particular script you can use the params command. This prints out the details of the
script, including the inputs.

$ wjm:examples will$ omero script params 301
Using session 1202acc0-4424-4fa2-84fe-7c9e069d3563 (root@localhost:4064). Idle timeout: 10.0 min. Current group: system
id: 301
name: Edit_Descriptions.py
version:
authors:
institutions:
description: Edits the descriptions of multiple Images, either specified via Image IDs or by the Dataset IDs.
See http://www.openmicroscopy.org/site/support/omero4/developers/scripts/user-guide.html for the tutorial that uses this script.

namespaces:
stdout: text/plain
stderr: text/plain
inputs:
New_Description - The new description to set for each Image in the Dataset
  Optional: True
  Type: ::omero::RString
  Min:
  Max:
  Values:
IDs - List of Dataset IDs or Image IDs
  Optional: False
  Type: ::omero::RList
  Subtype: ::omero::RLong
  Min:
  Max:
  Values:
Data_Type - The data you want to work with.
  Optional: False
  Type: ::omero::RString
  Min:
  Max:
  Values: Dataset, Image
outputs:

13.2.3 Debugging scripts

The stderr and stdout from running a script should always be returned to you, either when running scripts from the command line, via OMERO.insight or using the scripts API. This should allow you to debug any problems you have.

You can also look at the output from the script in the OriginalFile directory, commonly stored in /OMERO/File/. The script file when executed is uploaded as a new OriginalFile, and the standard error, standard out are saved as the next two OriginalFiles after that. These files can be opened in a text editor to examine contents.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

13.3 Guidelines for writing OMERO.scripts

These guidelines for writing OMERO scripts are designed to improve the interaction of the scripts with OMERO clients so that they can:

• generate a nice, usable UI for the script
• handle the script results appropriately

If you want instructions on how to get started with OMERO scripts, see the link above or the OMERO.scripts user guide.

Most of the points below are implemented in the example Edit_Descriptions.py11.

13.3.1 Script naming and file path

• Script Name should be in the form ‘Script_Name.py’. OMERO.insight will replace underscores with spaces in the script selection menu.

11https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/ScriptingService/Edit_Descriptions.py
• File paths - OMERO.insight will use the parent folder to build a scripts menu, capitalising and removing underscores. For example, a script uploaded from /omero/export_scripts/Batch_Image_Export.py will be displayed in OMERO.insight under “Export Scripts”.

• Script Descriptions should give a brief summary of what the script does. If a longer description or instructions for using the script are desired, it is suggested that a URL is included. The description will be displayed in the script UI and any URLs will be ‘clickable’ to launch a browser.

13.3.2 Parameters

• Parameter Names should be in the form ‘Parameter_Name’. Underscores will be replaced by spaces in the UI generated in OMERO.insight.

• Where applicable, Parameters should be supplied with a list of options. For example:

```python
scripts.String("Algorithm", values=[rstring('Maximum_Intensity'),rstring('Mean_Intensity')])
```

• Where possible, parameters should be supplied with default values. These will be used to populate fields in the OMERO.insight script UI and will be used by default when launching the script from the command line.
OMERO Documentation, Release 5.2.6

13.3.3 Parameter grouping / ordering

Parameters are not ordered by default. They can be ordered and grouped by adding a “grouping” attribute, which is a string, where ‘groups’ are separated by a ‘.’ E.g. “01.A”. Parameters will be ordered by the lexicographic sorting of this string and groups indicated in the UI. In most cases this will simply be a common indentation of parameters in the same group. In addition, if the ‘parent’ parameter of a group is a boolean, then un-checking the check-box in the UI will disable the child parameters. For example a UI generated from the code below will have a ‘Show Scalebar’ option. If this is un-checked, then the Size and Colour parameters will be disabled and will not be passed to the script.

```python
scripts.Bool("Show_Scalebar", grouping="10", default=True),
scripts.Int("Scalebar_Size", grouping="10.1"),
scripts.String("Scalebar_Colour", grouping="10.2"),
```

13.3.4 Pick selected Images, Datasets or Projects from OMERO clients

Both OMERO.insight and OMERO.web recognize and populate a pair of fields named ‘Data_Type’ (string) and ‘IDs’ (Long list) with the objects currently selected in the client UI when the script is launched. You should specify the ‘Data_Type’ options that your script should accept. E.g.

```python
dataTypes = [rstring('Dataset'), rstring('Image')]
client = scripts.client('Thumbnail_Figure.py', "Export a figure of thumbnails",
    scripts.String("Data_Type", optional=False, grouping="01", values=dataTypes, default="Dataset"),
    scripts.List("IDs", optional=False, grouping="02").ofType(rlong(0))
)
```

13.3.5 Script outputs

- Scripts may return a short message to report success or failure. This should use the key: ‘Message’ in the output map. This will be displayed in OMERO.insight when the script completes.

```python
client.setOutput("Message", rstring("Script generated new Image"))
```

- Scripts that generate an Image should return the ImageI object. OMERO.insight will provide a link to view the Image. The key that is used (“Image” in this example) is not important for this to work, but ‘image’ should be an omero.model.ImageI object.

```python
client.setOutput("Image", robject(image))
```

- Scripts that generate a File Annotation or Original File should return these objects. OMERO.insight will give users the option of downloading the File, and may also allow viewing of the file if it is of a suitable type. This should be set as the mimetype of the File Annotation (E.g. ‘plain/text’, ‘image/jpeg’ etc). In this example, fileAnnotation should be an omero.model.FileAnnotationI object, but could also be an omero.model.OriginalFileI object.
13.3.6 More tips

- Use the ‘unwrap()’ function from omero.rtypes to unwrap rtypes from the script parameters since this function will iteratively unwrap lists, maps etc.

```python
from omero.rtypes import *
scriptParams = {}
for key in client.getInputKeys():
    if client.getInput(key):
        scriptParams[key] = unwrap(client.getInput(key))

print scriptParams  # stdout will be returned - useful for bug fixing etc.
```

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

13.4 MATLAB and Python

MATLAB functionality can be mixed into Python scripts via the Python package Mlabwrap\(^{12}\).

13.4.1 Installing Mlabwrap

To install MlabWrap follow the installation guide\(^{13}\) and make sure that the paths are set for the environment variables:

```bash
LD_LIBRARY_PATH=$MATLABROOT/bin/Platform
MLABRAW_CMD_STR=$MATLABROOT/bin/matlab
```

13.4.2 Example MATLAB scripts

Below are some sample scripts showing MATLAB being launched from OMERO.scripts. MATLAB functions can also call the OMERO Java language bindings interface to access the server from the MATLAB functions.

**Calling a simple MATLAB function**

```python
import omero, omero.scripts as script
# import mlabwrap to launch matlab.
from mlabwrap import matlab;
client = script.client("rand.py", "Get matrix of random numbers drawn from a uniform distribution",
    script.Long("x").inout(), script.Long("y").inout())
client.createSession()
x = client.getInput("x").val
y = client.getInput("y").val

# call the matlab rand function via mlabwrap will automatically launch matlab
```

---

\(^{12}\)http://mlabwrap.sourceforge.net

\(^{13}\)http://mlabwrap.sourceforge.net/#installation
# if it is not already running on the system and call the rand method.
val = matlab.rand(x,y);

print val

Using the OMERO interface inside MATLAB

This example shows the MATLAB script being called, passed to the client object and accessing the same client instance as the script.

```python
import omero, omero.scripts as script
# import mlabwrap to launch matlab.
from mlabwrap import matlab;
client = script.client("projection.py", "Call the matlab projection code",
script.String("iceConfig").in(), script.String("user").in(),
script.String("password").in(),
script.Long("pixelsId").inout(), script.String("method").inout()
script.Long("stack").inout())

client.createSession()

iceConfig = client.getInput("pixelsId").val
user = client.getInput("pixelsId").val
password = client.getInput("pixelsId").val
method = client.getInput("method").val
stack = client.getInput("stack").val;

image = matlab.performProjection(iceConfig, username, password, pixelsId, stack, method);
```

The MATLAB projection method

```matlab
function performProjection(iceConfig, username, password, pixelsId, zSection, method)

omerojavaService = createOmeroJavaService(iceConfig, username, password);
pixels = getPixels(omerojavaService, pixelsId);
stack = getPlaneStack(omerojavaService, pixelsId, zSection);
projectedImage = ProjectionOnStack(stack, method);

function [resultImage] = ProjectionOnStack(imageStack,type)

[zSections, X, Y] = size(imageStack);
if(strcmp(type,'mean') || strcmp(type, 'sum'))
    resultImage = squeeze(sum(imageStack));
    if(strcmp(type,'mean'))
        resultImage = resultImage./zSections;
    end
end
if(strcmp(type,'max'))
    resultImage = squeeze(max(imageStack,[],1));
end
```

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.
13.5 OMERO.scripts advanced topics

13.5.1 Regular user (non-admin) workflow

If you are using a server for which you do not have admin access, you must upload scripts as ‘user’ scripts, which are not trusted to run on the server machine. The OMERO scripting service will still execute these scripts in a similar manner to official ‘trusted’ scripts but behind the scenes it uses the client machine to execute the script. This means that any package imports required by the script should be available on the client machine.

The first step is to connect to the server and set up the processor on the client (see diagram, below).

![OMERO Scripting Workflow Diagram](image)

Figure 13.6: OMERO scripting workflow

- You need to download ‘Ice’ from ZeroC and set the environment variables, as described in the server installation page (see Unix and Windows versions).
- You also need the OMERO server download. Go to the OMERO downloads page and get the appropriate server package (version must match the server you are connecting to). Unzip the package in a suitable location.

In a command line terminal, change into the unzipped OMERO package, connect to the server and start user processor. For example for host: openmicroscopy.org.uk and user: will

```bash
$ cd Desktop/OMERO.server-Beta-4.2/
$ bin/omero -s openmicroscopy.org.uk -u will script serve user
$ password: ......
```

If you want to run scripts belonging to another user in the same collaborative group you need to set up your local user processor to accept scripts from that user. First, find the ID of the user, then start the user processor and give it the user’s ID:

```bash
$ cd Desktop/OMERO.server-Beta-4.2/
$ bin/omero -s openmicroscopy.org.uk -u will user list
$ bin/omero -s openmicroscopy.org.uk -u will script serve user=5
```

From this point on, the user and admin workflows are the same, except for a couple of options that are not available to regular users. Also see below.

*Note:* Because non-official scripts do not have a unique path name, you will be able to run the upload command multiple times on the same file. This will create multiple copies of a file in OMERO and then you will have to choose the most recent one (highest ID) if you want to run the latest script. It is best to avoid this and use the ‘replace’ command as for official scripts.

---

13.5. OMERO.scripts advanced topics

---

14 [http://downloads.openmicroscopy.org/latest/omero/]
To list user scripts:

```bash
$ omero -s openmicroscopy -u will script list user  # lists user scripts
```

<table>
<thead>
<tr>
<th>id</th>
<th>Scripts for user</th>
</tr>
</thead>
<tbody>
<tr>
<td>151</td>
<td>examples/HelloWorld.py</td>
</tr>
<tr>
<td>251</td>
<td>examples/Edit_Descriptions.py</td>
</tr>
</tbody>
</table>

You can list scripts belonging to another user that are available for you (e.g. you are both in the same collaborative group) by using the user ID as described above:

```bash
$ omero user list
$ omero script list user=5
```

User scripts can be run from OMERO.insight. They will be found under ‘User Scripts’ in the scripts menu. Remember, for user scripts you will need to have the User-Processor running.

### 13.5.2 The iScript service

The OMERO.blitz server provides a service called [iScript](http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/IScript.html) that includes methods to upload, delete, query and run scripts. To access these methods a session needs to be created and the script service started. However, you may find it more convenient to use the command line `bin/omero script` or the OMERO.insight client to work with scripts as described on the [OMERO.scripts user guide](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/ScriptingService/adminWorkflow.py).

### 13.5.3 Scripting service API

The recommended way of working with the scripting service is via the command line as described on the [OMERO.scripts user guide](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/ScriptingService/adminWorkflow.py) page. The information on this page is only useful if you want to access the Scripting service from your own client-side Python code.

OMERO clients can upload, edit, list and run scripts on the OMERO server using the Scripting Service API. These methods (discussed below) are implemented in `examples/ScriptingService/adminWorkflow.py`16. This sample script allows these functions to be called from the command line and can be used as an example for writing your own clients.

Most functions of the `adminWorkflow.py` script are also implemented in the OMERO CLI described on the [OMERO.scripts user guide](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/ScriptingService/adminWorkflow.py), which is the preferred way of accessing the scripting service for script writers.

Having downloaded `examples/ScriptingService/adminWorkflow.py`17, you can get some instructions for using the script by typing:

```bash
$ python adminWorkflow.py help
```

To upload ‘official’ scripts, use the `uploadOfficialScript` method of the scripting service or use the `upload` command from `adminWorkflow.py` (you can omit password and enter it later if you do not want it showing in your console):

```bash
$ python adminWorkflow.py -s server -u username -p password -f script/file/to/upload.py upload
```

Official scripts must have unique paths. Therefore, the `uploadOfficialScript` method will not allow you to overwrite and existing script. However, the `adminWorkflow.py upload` command will automatically use `scriptService.editScript()` if the file exists. If you want to change this behavior, edit the `adminWorkflow.py` script accordingly.

To get the official scripts available to run, use the `getScripts()` method, which returns a list of Original Files (scripts). This code will produce a list of scripts like the one above.

---

16https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/ScriptingService/adminWorkflow.py
17https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/ScriptingService/adminWorkflow.py
scripts = scriptService.getScripts()
for s in scripts:
    print s.id.val, s.path.val + s.name.val

This can be called from adminWorkflow.py with this command:

$ python adminWorkflow.py -s server -u username -p password list

The script can then be run, using the script ID and passing the script a map of the input parameters. The adminWorkflow.py script has a `run` command that can be used to identify a script by its ID or path/name and run it. The `run` command will ask for parameter inputs at the command line.

$ python adminWorkflow.py -s localhost -u root -p omero -f scriptID run

or

$ python adminWorkflow.py -s localhost -u root -p omero -f omero/figure_scripts/Roi_Figure.py run

You can combine the latter form of this command with the ‘upload’ option to upload and run a script at once, useful for script writing and testing.

$ python adminWorkflow.py -s localhost -u root -p omero -f omero/figure_scripts/Roi_Figure.py upload run

Alternatively, you could edit adminWorkflow.py to ‘hard-code’ a set of input parameters for a particular script (this strategy is used by examples/ScriptingService/runHelloWorld.py\(^{1}\)). The code below shows a more complex example parameter map. This strategy might save you time if you want to be able to rapidly run and re-run a script you are working on. Of course, it is also possible to run scripts from OMERO.insight!

cNamesMap = omero.rtypes.rmap({
    '0': omero.rtypes.rstring("DAPI"),
    '1': omero.rtypes.rstring("GFP"),
    '2': omero.rtypes.rstring("Red"),
    '3': omero.rtypes.rstring("ACA")
})
blue = omero.rtypes.rstring('Blue')
red = omero.rtypes.rstring('Red')
mrgdColoursMap = omero.rtypes.rmap({
    '0': blue, '1': blue, '3': red
})
map = {
    "Image_IDs": omero.rtypes.rlist(imageIds),
    "Channel_Names": cNamesMap,
    "Split_Indexes": omero.rtypes.rlist([omerobj.rtypes.rlong(1), omerobj.rtypes.rlong(2)]),
    "Split_Panels_Grey": omero.rtypes.rbool(True),
    "Merged_Colours": mrgdColoursMap,
    "Merged_Names": omero.rtypes.rbool(True),
    "Width": omero.rtypes.rint(200),
    "Height": omero.rtypes.rint(200),
    "Image_Labels": omero.rtypes.rstring("Datasets"),
    "Algorithm": omero.rtypes.rstring("Mean_Intensity"),
    "Stepping": omero.rtypes.rint(1),
    "Scalebar": omero.rtypes.rint(10), # will be ignored since no pixelsize set
    "Format": omero.rtypes.rstring("PNG"),
    "Figure_Name": omero.rtypes.rstring("splitViewTest"),
    "Overlay_Colour": omero.rtypes.rstring("Red"),
    "ROI_Zoom": omero.rtypes.rfloat(3),
    "ROI_Label": omero.rtypes.rstring("fakeTest"), # won't be found - but should still work
}

\(^{1}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/ScriptingService/runHelloWorld.py
The results returned from running the script can be queried for script outputs, including stdout and stderr. The following code queries the results for an output named 'Message' (also displayed by OMERO.insight)

```python
print results.keys()
if 'Message' in results:
    print results['Output_Message'].getValue()
if 'stdout' in results:
    origFile = results['stdout'].getValue()
    print "Script generated StdOut in file:", origFile.getId().getValue()
if 'stderr' in results:
    origFile = results['stderr'].getValue()
    print "Script generated StdErr in file:", origFile.getId().getValue()
```

This code has been extended in adminWorkflow.py to display any StdErr and StdOut generated by the script when it is run.
14.1 OMERO.web framework

OMERO.web is a framework for building web applications for OMERO. It uses Django\(^1\) to generate HTML pages from data retrieved from the OMERO server. OMERO.web acts as a Python client of the OMERO server using the OMERO API, as well as being a web server itself (see ‘infrastructure’ info below). It uses Django ‘apps’ to provide modular web tools, such as the webclient (working with image data, administrator management). This modular framework makes it possible to extend OMERO.web with your own apps.

\(^1\)https://www.djangoproject.com/
One of the apps, WebGateway, provides utility methods for accessing data and rendering images, as well as handling the connection to OMERO server.

### 14.1.1 OMERO.web infrastructure

#### OMERO Python API

The OMERO.web framework is all based on the OMERO Python API, using the Blitz Gateway (see [OMERO Python language bindings](http://downloads.openmicroscopy.org/latest/omero/api/python/omeroweb/omeroweb.webgateway.html#module-omeroweb.webgateway)). The OMERO.web framework provides functionality for creating and retrieving connections to OMERO (see example below and [Writing OMERO.web views](http://downloads.openmicroscopy.org/latest/omero/api/python/omeroweb/omeroweb.webgateway.html#module-omeroweb.webgateway))

#### Web gateway

The webgateway is a Django app that provides utility functionality for the other web components. This includes a full image viewer, as well as methods for rendering images etc. You can browse the webgateway URLs, or see the WebGateway page for more info.

#### Web apps

The OMERO.web framework consists of several Django apps denoted by folders named ‘web....’. These include webgateway, as discussed above, as well as released tools (webclient) and other apps in development:

- **webclient** Main web client for viewing images, annotating, administration of user and group settings etc. More information available under [OMERO.web](http://downloads.openmicroscopy.org/latest/omero/api/python/omeroweb/omeroweb.webgateway.html#module-omeroweb.webgateway)
- **webgateway** A web services interface, providing rendered images and data. See [WebGateway](http://downloads.openmicroscopy.org/latest/omero/api/python/omeroweb/omeroweb.webgateway.html#module-omeroweb.webgateway).

**Note:** Although it is possible to access functionality from any installed app, **ONLY webgateway** should be considered as a stable public API. URLs and methods within other web apps are purely designed to provide internal functionality for the app itself and may change in minor releases without warning.

Additional apps can be easily added to your OMERO.web deployment. One example is the webtest app that contains several code samples mentioned in the following pages. You will find install instructions on the webtest app page itself.

### 14.1.2 Getting started

The preferred workflow for extending OMERO.web is to create a new Django app. Django apps provide a nice way for you to keep all your code in one place and make it much easier to port your app to new OMERO releases or share it with other users. To get started, see [Creating an app](http://downloads.openmicroscopy.org/latest/omero/api/python/omeroweb/omeroweb.webgateway.html#module-omeroweb.webgateway). Further documentation on editing the core OMERO.web code is at [Editing OMERO.web](http://downloads.openmicroscopy.org/latest/omero/api/python/omeroweb/omeroweb.webgateway.html#module-omeroweb.webgateway). If you want to have a quick look at some example code, see below.

**Quick example - webtest**

This tiny example gives you a feel for how the OMERO.web framework gets data from OMERO and displays it on a web page. You can find this and other examples in the webtest app.

There are 3 parts to each page: url, view and template. For example, this code below is for generating an HTML page of a Dataset (see screen-shot). If you have OMERO.web running, you can view the page under `http://servername.example.org/webtest/dataset/<datasetId>`. If you have OMERO.web running, you can view the page under `http://servername.example.org/webtest/dataset/<datasetId>`.

- **url** goes in omeroweb/webtest/urls.py This maps the URL ‘webtest/dataset/<datasetId>/’ to the View function ‘dataset’, passing it the datasetId.

---


3.[https://github.com/openmicroscopy/webtest/](https://github.com/openmicroscopy/webtest/)

4.[https://github.com/openmicroscopy/webtest/](https://github.com/openmicroscopy/webtest/)
**View** function, in omeroweb/webtest/views.py. NB: @login_required decorator retrieves connection to OMERO as ‘conn’ passed in args to method. See *Writing OMERO.web views* for more details.

```python
from omeroweb.webclient.decorators import login_required
# handles login (or redirects). Was @isUserConnected before OMERO 4.4
@login_required
def dataset(request, datasetId, conn=None, **kwargs):
    ds = conn.getObject("Dataset", datasetId)
    # generate html from template
    return render(request, 'webtest/dataset.html', {'dataset': ds})
```

**Template:** The template webpage, in omeroweb/webtest/templates/webtest/dataset.html

```html
<html>
<body>
<h1>{{ dataset.getName }}</h1>
{% for i in dataset.listChildren %}
<div style="float:left; padding:10px">
    <img src="{% url 'webgateway.views.render_thumbnail' i.id %}" />
    <br />
    {{ i.getName }}
</div>
{% endfor %}
</body>
</html>
```
OMERO Documentation, Release 5.2.6

- Next: Get started by OMERO.web deployment for developers....

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

14.2 OMERO.web deployment for developers

14.2.1 Getting set up

You will need to have an OMERO server running that you can connect to. This will typically be on your own machine, although you can also connect to an external OMERO server. You can add the server to the `omero.web.server_list` and choose that server when you log in. You should enable `omero.web.debug` and start a lightweight development Web server on your local machine.

Note: Since OMERO 5.2, the OMERO.web framework no longer bundles a copy of the Django package, instead manual installation of the Django dependency is required. It is highly recommended to use Django 1.8\(^5\) (LTS) which requires Python 2.7. For more information see Python on the Version requirements page.

14.2.2 Using the lightweight development server on Unix

All that is required to use the Django lightweight development server is to set the `omero.web.application_server` configuration option, turn `omero.web.debug` on and start the server up:

```
$ bin/omero config set omero.web.application_server development
$ bin/omero config set omero.web.debug True
$ bin/omero web start
INFO:__main__:Application Starting...
INFO:root:Processing custom settings for module omeroweb.settings
... Validating models...
0 errors found
Django version 1.8, using settings 'omeroweb.settings'
Starting development server at http://127.0.0.1:4080/
Quit the server with CONTROL-C.
```

14.2.3 Using the lightweight development server on Windows

All that is required to use the Django lightweight development server is to set the `omero.web.application_server` configuration option, turn Debugging on and start the server up:

```
C:\omero_dist>bin\omero config set omero.web.application_server development
C:\omero_dist>bin\omero config set omero.web.debug True
C:\omero_dist>bin\omero web start
INFO:__main__:Application Starting...
INFO:root:Processing custom settings for module omeroweb.settings
... Validating models...
0 errors found
Django version 1.8, using settings 'omeroweb.settings'
Starting development server at http://127.0.0.1:4080/
Quit the server with CTRL-BREAK.
```

\(^5\)https://docs.djangoproject.com/en/1.8/releases/1.8/
14.2.4 Using WSGI (Unix/Linux)

For convenience you may wish to run a web server under your local user account instead of using a system server for testing. Install Nginx and Gunicorn (See OMERO.web deployment) but generate a configuration file using the following commands:

```bash
$ bin/omero config set omero.web.application_server 'wsgi-tcp'
$ bin/omero web config nginx-development > nginx-development.conf
```

Start Nginx and the Gunicorn worker processes running one thread listening on 127.0.0.1:4080 that will autoreload on source change:

```bash
$ nginx -c $PWD/nginx-development.conf
$ bin/omero config set omero.web.application_server.max_requests 1
$ bin/omero config set omero.web.wsgi_args ---*--reload*
$ bin/omero web start
```

Next: Get started by Creating an app....

---

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

14.3 Creating an app

The Django web site has a very good tutorial\(^6\) to get you familiar with the Django framework. The more you know about Django, the easier you will find it working with the OmeroWeb framework. One major feature of Django that we do not use in OmeroWeb is the Django database mapping, since all data comes from the OMERO server and is saved back there. You will notice that the models.py files in each app are empty.

14.3.1 Getting set up

In order to deploy OMERO.web in a development or testing environment please follow the instructions under OMERO.web deployment for developers.

You should make sure that you can access the webclient on your local machine before starting to develop your own code. Be sure to use the correct port number, e.g.:

- http://localhost:4080/webclient/

When you edit and save your app, Django will automatically detect this and you only need to refresh your browser to see the changes.

If you want to make changes to the OMERO.web code itself, you should see Editing OMERO.web.

You can place your app anywhere on your PYTHONPATH, as long as it can be imported by OMERO.web.

14.3.2 Creating an app

We suggest you use GitHub\(^7\) (as we do) since it is much easier for us to help you with any problems you have if we can see your code. The steps below describe how to create a stand-alone git repository for your app, similar to webtagging\(^8\). If you do not want to use GitHub, simply ignore the steps related to GitHub.

The steps below describe how to set up a new app. You should choose an appropriate name for your app and use it in place of <your-app>:

---

\(^6\)https://docs.djangoproject.com/en/1.8/intro/tutorial01/
\(^7\)https://github.com/
\(^8\)https://github.com/MicronOxford/webtagging
Add your app to your PYTHONPATH

Your app needs to be within a directory that is on your PYTHONPATH. We usually create a new container for a new app, and add it to the PYTHONPATH.

$ mkdir PARENT-APP-DIR
$ cd PARENT-APP-DIR
$ export PYTHONPATH=$PYTHONPATH:/path/to/PARENT-APP-DIR

OR you could simply choose an existing location:

$ cd /somewhere/on/your/pythonpath/

Create and check out a new GitHub repository OR manually create a new directory

- Login to your GitHub account homepage (e.g. https://github.com/<your-name>/) and click “New repository”
- Enter the name of <your-app>, add a description and choose to add a README.
- Check out your new repository (into a new directory)

$ git clone git@github.com:<your-name>/<your-app>.git

- OR: If you have not used git to create your app directory above, then

$ mkdir <your-app>

- In either case, you should now have a directory called <your-app> within a directory that is on your PYTHONPATH.

Add the essential files to your app

- Create an empty file <your-app>/__init__.py (NB: both double underscores)
- Create urls.py

```python
from django.conf.urls import *
from <your-app> import views

urlpatterns = patterns('django.views.generic.simple',
    # index 'home page' of the <your-app> app
    url(r'^$', views.index, name='<your-app>_index'),
)
```

- Create views.py

```python
from django.http import HttpResponse

def index(request):
    ""
    Just a place-holder while we get started
    ""
    return HttpResponse("Welcome to your app home-page!")
```

14.3. Creating an app 400
For more details on how to write views, forms, using templates, etc. check the Django documentation\(^9\).

**Add your app to OMERO.web**

`omeo.web.apps` adds your custom application to the `INSTALLED_APPS`, so that URLs are registered etc.

**Note:** Here we use single quotes around double quotes, since we are passing a double-quoted string as a json object.

```bash
$ bin/omero config append omero.web.apps '"<your-app>"'
```

Now you can view the home-page we created above (NB: you will need to restart the OMERO.web server for the config settings to take effect).

```bash
$ bin/omero web stop
$ bin/omero web start
```


**Commit your code and push to GitHub**

```bash
$ git status (see new files, plus .pyc files)
$ echo "*.pyc" > .gitignore # ignore .pyc files
$ echo ".gitignore" >> .gitignore  # ALSO ignore .gitignore

$ git add ./
$ git commit -m "Initial commit of bare-bones OMERO.web app"
$ git push origin master
```

**Connect to OMERO: example**

We have got our new app working, but it is not connecting to OMERO yet. Let us create a simple “stack preview” for an Image with multiple Z-sections. We are going to display the image name and 5 planes evenly spaced across the Z-stack. You should be able to add the appropriate code to `urls.py`, `views.py` that you created above, and add a template under `/omeroweb/<your-app>/templates/<your-app>/`

**Note:** note that /<your-app>/ appears twice in that path (need an extra folder under templates).

The following example can be found in the `webtest\(^{10}\)` app.

- **urls.py**

  ```python
  url( r'^stack_preview/(?P<imageId>[0-9]+)/$', views.stack_preview,
      name="<your-app>_stack_preview" ),
  ```

  ```python
  ```

- **views.py** Here we are using the `@login_required` decorator to retrieve a connection to OMERO from the session key in the HTTP request (or provide a login page and redirect here). `conn` is passed to the method arguments. NB: Note a couple of new imports to add at the top of your page.

\(^9\)https://docs.djangoproject.com/en/1.8/intro/tutorial03/#write-your-first-view

\(^{10}\)https://github.com/openmicroscopy/webtest/
from omeroweb.webclient.decorators import login_required
from django.shortcuts import render

@login_required()
def stack_preview(request, imageId, conn=None, **kwargs):
    """ Shows a subset of Z-planes for an image ""
    image = conn.getObject("Image", imageId)  # Get Image from OMERO
    image_name = image.getName()
    sizeZ = image.getSizeZ()  # get the Z size
    # 5 Z-planes
    z_indexes = [0, int(sizeZ*0.25),
                 int(sizeZ*0.5), int(sizeZ*0.75), sizeZ-1]
    return render(request, 'webtest/stack_preview.html',
                   {'imageId': imageId,
                    'image_name': image_name,
                    'z_indexes': z_indexes})

• `<your-app>/templates/<your-app>/stack_preview.html`

    <html>
    <head>
    <title>Stack Preview</title>
    </head>
    <body>
    <h1>{{ image_name }}</h1>
    {% for z in z_indexes %}
    <img src="{% url 'webgateway.views.render_image' imageId z 0 %}" style="max-width: 200px; max-height:200px"/>
    {% endfor %}
    </body>
    </html>

Viewing the page at http://localhost:4080/<your-app>/stack_preview/<image-id>/ should give you the image name and 5 planes from the Z stack. You will notice that we are using the webgateway to handle the image rendering using a URL auto-generated by Django - see WebGateway.

### 14.3.3 Resources for writing your own code

The webtest\(^\text{11}\) app has a number of examples. Once installed, you can go to the webtest homepage e.g. http://localhost:4080/webtest and you will see an introduction to some of them. This page tries to find random images and datasets from your OMERO server to use in the webtest examples.

### Using jQuery and jQuery UI from OMERO.web

OMERO.web uses the jQuery\(^\text{12}\) and jQuery UI\(^\text{13}\) javascript libraries extensively. If you need these libraries, you can include the OMERO.web versions of these libraries in your own pages. The alternative is to provide a specific version of jQuery or jQuery UI in your own app if, for example, you think that a version change may cause problems in your code. If you need to make use of these resources in your own pages, you can add the following statements to the `<head>` of your page templates:

<!-- jQuery -->
{% include "webgateway/base/includes/script_src_jquery.html" %}

---

\(^{11}\)https://github.com/openmicroscopy/webtest/
\(^{12}\)https://jquery.com/
\(^{13}\)https://jqueryui.com/
The OMERO 5.1.0 release included an upgrade to jQuery 1.11.1 and jQuery UI 1.10.4. We have added jQuery-migrate library (1.2.1) to reduce the effects of this upgrade. You should inspect the javascript console during development to observe any warning messages.

### Extending templates

We provide several HTML templates in webgateway/templates/webgateway/base. This is a nice way of giving users the feeling that they have not left the webclient, if you are providing additional functionality for webclient users. You may choose not to use this if you are building a ‘stand-alone’ web application. In either case, it is good practice to create your own templates with common components (links, logout etc.), so you can make changes to all your pages at once. See Writing page templates in OMERO.web for more info.

### 14.3.4 App settings

You can add settings to your app that allow configuration via the command line in the same way as for the base OMERO.web. The list of CUSTOM_SETTINGS_MAPPINGS in components/tools/OmeroWeb/omeroweb/settings.py is a good source for examples of the different data types and parsers you can use.

For example, if you want to create a user-defined setting yourapp.foo, that contains a dictionary of key-value pairs, you can add to CUSTOM_SETTINGS_MAPPINGS in yourapp/settings.py:

```python
import json
CUSTOM_SETTINGS_MAPPINGS = {
    "omero.web.yourapp.foo": ["FOO", {'key': "val"}, json.loads]
}
```

From somewhere else in your app, you can then access the settings:

```python
from yourapp import settings
print settings.FOO
```

Users can then configure this on the command line as follows:

```
$ bin/omero config set omero.web.yourapp.foo '{"userkey": "userval"}'
```

### OMERO.web top links

You can configure omero.web.ui.top_links to add links to the list of links at the top of the webclient main pages.

- **Name your URL in urls.py** (optional). Preferably we use URL names to refer to URLs. For example, the homepage of your app might be named like this in yourapp/urls.py.

  ```python
  url(r'^$', views.index, name='figure_index'),
  ```

  You can then refer to the link defined above using this name, or you can simply use a full URL for external links.

- **Update configuration** Use the OMERO command line interface to append the link to the top_links list.

  Links use the format ["Label", "URL_name"] or you can follow this example:

  ```python
  ["Add a new project", "app_new_project"]
  ```

---

14[https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/tools/OmeroWeb/omeroweb/settings.py](https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/tools/OmeroWeb/omeroweb/settings.py)
Custom image viewer

If you have created your own image viewer and would like to have it replace the existing image viewer in the webclient, this can be configured using `omero.web.viewer.view`.

You will need your `views.py` method to take an Image ID with a parameter named `iid`. For example, see `channel_overlay_viewer` from the `webtest` app:

```python
@login_required()
def channel_overlay_viewer(request, iid, conn=None, **kwargs):
    # Your implementation

@bin/omero config set omero.web.viewer.view webtest.views.channel_overlay_viewer
```

You can then configure the webclient to use this viewer by providing the full path name to this view method. For example, if you have `webtest` installed you can use the `channel_overlay_viewer`:

```bash
$ bin/omero config set omero.web.viewer.view webtest.views.channel_overlay_viewer
```

This will now direct the image viewer url at `webclient/img_detail/<iid>/` to this viewer. However, the existing viewer will still be available under webgateway at `webgateway/img_detail/<iid>/`.

If you want to use a different viewer for different images, you can conditionally redirect to the webgateway viewer or elsewhere. For example:

```python
if image.getSizeC() == 1:
    return HttpResponseRedirect(reverse("webgateway.views.full_viewer", args=(iid,)))
```

14.3.5 OMERO.web plugins

If you want to display content from your app within the webclient UI, please see [Webclient Plugins](#webclient-plugins).

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

14.4 Webclient Plugins

The webclient UI can be configured to include content from other web apps. This allows you to extend the webclient UI with your own functionality. This is used by the `webtagging` app[16] and there are also some examples in the `webtest`[17] app.

Currently you can add content in the following locations:

- **Center Panel** Adding a panel to the center of the webclient will display a drop-down menu to the top right of the center panel, allowing users to choose your plugin.

---

[17]https://github.com/openmicroscopy/webtest/
• **Right Panel** You can add additional tabs to the right panel. These will be available in the main webclient page as well as history and search result pages.

### 14.4.1 Overview

To begin with, you need to prepare your plugin pages in your own app, with their own URLs, views and templates. Then you can display these pages within the webclient UI, using the plugin framework.

**Note:** When you embed your pages in the webclient, there is potential for conflicts between JavaScript and CSS in the host page and your own code. Care must be taken not to overwrite global JavaScript functions (such as jQuery or the ‘OME’ namespace) or to add CSS codes that may affect host elements.

The webclient plugins work by adding some custom JavaScript snippets into the main pages of the webclient and adding HTML elements to specified locations in the webclient. These snippets of JavaScript can be used to load content into these HTML elements. Usually you will want to do this dynamically, to display data based on the currently selected objects (although this is optional). Helpers can be used to respond to changes in the selected objects and the selected tab, so you can load or refresh your plugin only when necessary.

### 14.4.2 App URLs

To display content based on currently selected data, such as Projects, Datasets and Images, your app pages will need to have these defined in their URLs. For example:

```python
# Webtagging: Tag images within the selected dataset
url( r'^auto_tag/dataset/(?P<datasetId>[0-9]+)/$', views.auto_tag ),

# Webtest: Show a panel of ROI thumbnails for an image
url( r'^image_rois/(?P<imageId>[0-9]+)/$', views.image_rois ),
```

These URLs should simply display the content that you want to show in the webclient. NB: when these pages load in the webclient, they will have all the webclient CSS and JavaScript (such as jQuery) available so you do not need to include these in your page.

### 14.4.3 Configuring the plugin

**Choose an element ID**

You will need to specify an ID for the `<div>` element that is added to the webclient, so that you can refer to this element in the JavaScript. For example, “image_roi_tab” or “auto_tag_panel”.

**Create a JavaScript file**

This will contain the JavaScript snippet that is injected into the main webclient page `<head>` when the page is generated. This is added using Django’s templates, so it should be placed within your app’s `/templates/<app-name>/` directory and named `.html`, e.g. `/templates/<app-name>/webclient_plugins/right_plugin_rois.html`. All the JavaScript should be within `<script>` and `</script>` tags. Your plugin initialization should happen after the page has loaded, so you use the jQuery on-ready function.

You use custom jQuery functions, called ‘omeroweb_right_plugin’ or ‘omeroweb_center_plugin’, to initialize the webclient plugin. These will handle all the selection change events. You simply need to specify how the panel is loaded, based on the selected object(s) and what objects are supported. The plugin will be disabled when non-supported objects are selected.

Below is a simple example of their usage. More detailed documentation available in the **plugin options section** below.

**Center Panel Plugin**
14.4.4 Plugin installation

Now you need to add your plugin to the appropriate plugin list, stating the displayed name of the plugin, the path/to/js_snippet.html and the ID of the plugin element. Plugin lists are:

- omero.web.ui.center_plugins
- omero.web.ui.right_plugins

Use the OMERO command line interface to add the plugin to the appropriate list.
The right_plugins list includes the Acquisition tab and Preview tab by default. If you want to append the webtest ROI plugin or your own plugin to the list, you can simply do:

```bash
$ bin/omero config append omero.web.ui.right_plugins
'[
  "ROIs", "webtest/webclient_plugins/right_plugin.rois.js.html", "image_roi_tab"
]
```

If you want to replace existing plugins and display only your own single plugin, you can simply do:

```bash
$ bin/omero config set omero.web.ui.right_plugins
'[
  "ROIs", "webtest/webclient_plugins/right_plugin.rois.js.html", "image_roi_tab"
]
```

### Restart Web

Stop and restart your web server, then refresh the webclient UI. You should see your plugin appear in the webclient UI in the specified location. You should only be able to select the plugin from the drop-down menu or tab if the supported data type is selected, e.g. ‘image’. When you select your plugin, the load content method you specified above will be called and you should see your plugin loaded.

### Refreshing content

If you now edit the `views.py` or HTML template for your plugin and want to refresh the plugin within the webclient, all you need to do is to select a different object (e.g. dataset, image etc). If you select an object that is not supported by your plugin, then nothing will be displayed, and for the right-tab plugin, the tab selection will change to the first tab.

### 14.4.5 Plugin options

- **supported_obj_types**: If your plugin displays data from single objects, such as a single Image or Dataset, you can specify that here, using a list of types:

  ```javascript
  supported_obj_types: ['dataset', 'image'],
  ```

  This will ensure that the plugin is only enabled when a single Dataset or Image is selected. To support multiple objects, see `tab_enabled`.

- **plugin_enabled**: This function allows you to specify whether a plugin is enabled or not when specified objects are selected. It is only used if you have NOT defined `supported_obj_types`. The function is passed a single argument:

  ```javascript
  plugin_enabled: function(selected){
  if (selected.length == 0) return false;
  var dtype = selected[0]['id'].split('-')[0];
  if (selected.length > 1) {
    return (dtype == "image");
  } else {
    return ($.inArray(dtype, ['image', "dataset"]) > -1);
  }
  }
  ```
• **load_plugin_content** / **load_tab_content**: This function will be called when the plugin/tab content needs to be refreshed, either because the plugin is displayed for the first time, or because the selected object changes. The function will be passed 3 arguments:
  - `selected`: This is a list of the selected objects e.g. `[{'id': 'image-123'}, {'id': 'image-456'}]`
  - `obj_drtype`: This is the data-type of the first selected object, e.g. `image`
  - `obj_id`: This is the ID of the first selected object, e.g. `123`

### 14.4.6 OMERO 4.4.x backwards compatibility

If an OMERO 5 plugin also needs to work for OMERO 4.4.x installations, there are three things to consider:

#### URLs in templates

The `url` tag\(^{18}\) has changed in Django 1.8\(^{19}\) from `% url myview %` to `{{ url 'myview' }}`. In order for your OMERO 4.4.x install to understand the new syntax, it is simply a matter of adding `% load url from future %` to the top of the template file. By using the new-style syntax and the `% load url from future %` the template will be valid in both OMERO 4.4.x and OMERO 5. The `% load url from future %` line does no harm when used on Django 1.8\(^{20}\).

#### URL imports

The import statement required to get the URL and patterns objects in the plugin’s `urls.py` has changed as well. The following import can be used to import from the correct place depending on which Django version is present.

```python
import django
if django.VERSION < (1, 6):
    from django.conf.urls.defaults import *
else:
    from django.conf.urls import *
```

#### Plugin settings

In OMERO 4.4.x (with the older Django) it was possible to manipulate settings in the plugin’s `settings.py` like so:

```python
from django.conf import settings

# We can directly manipulate the settings
# E.g. add plugins to CENTER_PLUGINS list
settings.CENTER_PLUGINS.append(
    ["Auto Tag", "webtagging/auto_tag_init.js.html", "auto_tag_panel"]
)
```

This is no longer possible in OMERO 5 with Django 1.8\(^{21}\) and it is necessary to use `omero config set` to change this setting as documented here: [Plugin Installation](https://docs.djangoproject.com/en/1.8/releases/1.8/).

In order to preserve the old functionality on OMERO 4.4.x servers, the plugin’s `settings.py` can be changed to:

```python
import django
if django.VERSION < (1, 6):
    from django.conf import settings
```

---

\(^{18}\)https://docs.djangoproject.com/en/1.8/ref/templates/builtins/#url
\(^{19}\)https://docs.djangoproject.com/en/1.8/releases/1.8/
\(^{20}\)https://docs.djangoproject.com/en/1.8/releases/1.8/
\(^{21}\)https://docs.djangoproject.com/en/1.8/releases/1.8/
This will only take effect on OMERO 4.4.x servers so remember to document that with OMERO 5 the config will need to be modified by the person doing the installation.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 14.5 Editing OMERO.web

If you need to make changes to OMERO.web itself, then you will want to check out the OMERO source code. You can directly edit and run the OMERO.web code. This means that you benefit from the convenience of editing, saving and refreshing your browser without any build step.

However, you will still need to build OMERO (or download the release build) and set up your PYTHONPATH as described in the install documentation in order that you have the various dependencies such as Django.

You will then have 2 copies of the OMERO.web code - source code under components/tools/OmeroWeb/omeroweb and the server build under dist/lib/python/omeroweb.

To set up and run OMERO.web from the source code, you need to follow a few steps (commands are shown below):

- Set OMERO_HOME, so that OMERO.web knows where to find config, write logs etc.

  **Warning:** You should not set OMERO_HOME on production servers.

  The examples below assume a developer environment, making appropriate use of OMERO internals. See [Setting the OMERO_HOME environment variable](#) for details.

  - Make sure that the Django libraries that are under the build: dist/lib/python/django are on your PYTHONPATH.
  - Make sure the OmeroWeb folder: components/tools/OmeroWeb is on your PYTHONPATH.
  - Remove the built omeroweb folder, otherwise this will get used instead of the source omeroweb

  **Note:** You have to do this again if you build the server

- From the source omeroweb/ folder, manually run the Django development server

  ```
  # Example path to build target or downloaded directory
  $ export OMERO_HOME = ~/Desktop/OMERO/dist

  # Make sure the Django code etc can be imported
  $ export PYTHONPATH=$OMERO_HOME/lib/python/:$OMERO_HOME/../components/tools/OmeroWeb:$PYTHONPATH

  $ cd $OMERO_HOME
  # need to remove the built omeroweb code so it doesn’t get imported
  $ rm -rf lib/python/omeroweb/

  $ cd ../components/tools/OmeroWeb
  $ python omeroweb/manage.py runserver
  Validating models...
  0 errors found
  December 05, 2013 - 13:39:26
  Django version 1.8, using settings 'omeroweb.settings'
  ```
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.

Note: Default port number is 8000. To specify port, use E.g: $ python manage.py runserver 0.0.0.0:4080

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

14.6 WebGateway

WebGateway is a Django app within the OMERO.web framework. It provides a web API for rendering images and accessing data on the OMERO server via URLs.

Note: The OMERO.web client also supports URLs linking to specified data in OMERO. See the OMERO.web user guides\(^ {22}\) for more details.

14.6.1 Web services

This list of URLs below may be incomplete or out of date. For a complete list of URLs, see the latest API\(^ {23}\) and try the URLs out for yourself!

The HTTP request will need to include login details for creating or using a current server connection. This will be true for any request made after logging in to the server, e.g. login using the webclient login page then go to webgateway/... or if you have logged in to a server at http://ome.example.com/webclient then go to, for example, http://ome.example.com/webgateway/render_image/<imageid>/<z>/<t>/

![Figure 14.1: Rendered thumbnail](image)

URLs from within OMERO.web

Images rendered within OMERO.web templates should use Django’s {\% url \%} tag to generate URLs for webgateway, passing in the ID of the image. This is shown for each of the URLs below:

**Image viewer**

• Provides a full image viewer, with controls for scrolling Z and T, editing rendering settings etc.

URL: https://your_host/webgateway/img_detail/<imageid>/

Template tag: {\% url ‘webgateway.views.full_viewer’ image_id \%}

\(^ {22}\)http://help.openmicroscopy.org/

\(^ {23}\)http://downloads.openmicroscopy.org/latest/omero/api/python/omeroweb/omeroweb.webgateway.html#module-omeroweb.webgateway.urls
Images

- Returns a jpeg of the specified plane with current rendering settings

  URL: https://your_host/webgateway/render_image/<imageId>/?z=<z>/<t>/
  Template tag: {% url 'webgateway.views.render_image' image_id theZ theT %}

  From OMERO 4.4.4, omitting Z and T will use the default values:

  URL: https://your_host/webgateway/render_image/<imageId>/
  Template tag: {% url 'webgateway.views.render_image' image_id %}

- Makes a jpeg laying out each active channel in a separate panel

  URL: https://your_host/webgateway/render_split_channel/<imageId>/?z=<z>/<t>/
  Template tag: {% url 'webgateway.views.render_split_channel' image_id theZ theT %}

- Plots the intensity of a row of pixels in an image. w is line width

  URL: https://your_host/webgateway/render_row_plot/<imageId>/?z=<z>/<t>/<y>/<w>
  Template tag: {% url 'webgateway.views.render_row_plot' image_id theZ theT yPos width %}

- Plots the intensity of a column of pixels in an image.

  URL: https://your_host/webgateway/render_col_plot/<imageId>/?z=<z>/<t>/<x>/<w>/
  Template tag: {% url 'webgateway.views.render_col_plot' image_id theZ theT xPos width %}

- Returns a jpeg of a thumbnail for an image. w and h are optional (default is 64). Specify just one to retain aspect ratio. Since OMERO 5.1 it is also possible to specify Z and T indices in the query string.

  URL: https://your_host/webgateway/render_thumbnail/<imageId>/?z=10
  Template tag: {% url 'webgateway.views.render_thumbnail' image_id %}?z=10 # default size, z=10
  URL: https://your_host/webgateway/render_thumbnail/<imageId>/<w>/<h>
  Template tag: {% url 'webgateway.views.render_thumbnail' image_id 100 %} # size 100

Rendering settings

If no rendering settings are specified (as above), then the current rendering settings will be used. To apply different settings to images returned by the render_image and render_split_channels URLs, parameters can be specified in the request.

N.B. These settings are only applied to the rendered image and will not be saved unless specified.

Individual parameters are:

- **Channels on/off** e.g. for a 4 channel image, to turn on all channels except 2:
# From OMERO 4.4.4 you can simply specify the active channels
?
c=3 # only Channel 3 is active
?
c=3,4 # Channels 3 and 4 are active

- **Channel colour** e.g. to set the colours for channels 1 to red and 2 to green and 3 to blue:

?
c=1|\$FF0000,2|\$00FF00,3|\$0000FF

- **Rendering levels** e.g. to set the cut-in and cut-out values for a 3 Channel image.

?
c=1|400:505,2|463:2409,3|620:3879 # First channel inactive "-1"
?
c=2|463:2409,3|620:3879 # OMERO 4.4.4 only: inactive channels can be omitted

- **Z-projection**: Maximum intensity, Mean intensity or None (normal). By default we use all z-sections, but a range can be specified.

?
p=intmax
?
p=intmax|0:10 # Use z-sections 0-10 inclusive
?
p=intmean
?
p=normal

- **Rendering ‘Mode’**: greyscale or colour.

?
m=g # greyscale (only the first active channel will be shown in grey)
?
m=c # colour

- Parameters can be combined, e.g.

https://your_host/webgateway/render_image/2602/10/0/?c=1|100:505\$00550000FF,2|463:2409\$00FF00,3|620:3879

**JSON methods**

- **List of projects**: webgateway/proj/list/

  ```json
  [{
    "description": "",
    "id": 269,
    "name": "Aurora"
  },
  {
    "description": "",
    "id": 269,
    "name": "Drugs"
  }]
  ```

- **Project info**: webgateway/proj/<projectId>/detail/

  ```json
  {
    "description": "",
    "type": "Project",
    "id": 269,
    "name": "CenpA"
  }
  ```

- **List of Datasets in a Project**: webgateway/proj/<projectId>/children/

  ```json
  [{
    "child\_count": 9,
    "description": "",
    "type": "Dataset",
    "id": 270,
    "name": "Control"
  }]
  ```

- **Dataset, same as for Project**: webgateway/dataset/<datasetId>/detail/

- **Details of Images in the dataset**: webgateway/dataset/<datasetId>/children/
• Lots of metadata for the image. See below: webgateway/imgData/<imageId>/

Saving etc

• webgateway/saveImgRDef/<imageId>/
• webgateway/compatImgRDef/<imageId>/
• webgateway/copyImgRDef/

ImgData

The following is sample JSON data generated by /webgateway/imgData/<imageId>/

```
{
  "split_channel": {
    "c": {"width": 1448, "gridy": 2, "border": 2, "gridx": 3, "height": 966},
    "g": {"width": 966, "gridy": 2, "border": 2, "gridx": 2, "height": 966}
  },
  "rdefs": {"defaultValue": 0, "model": "color",
              "projection": "normal", "defaultZ": 15},
  "pixel_range": [-32768, 32767],
  "channels": [
    {"color": "0000FF", "active": true,
     "window": {"max": 449.0, "end": 314, "start": 70, "min": 51.0},
     "emissionWave": "DAPI",
     "label": "DAPI"},
    {"color": "00FF00", "active": true,
     "window": {"max": 7226.0, "end": 1564, "start": 396, "min": 37.0},
     "emissionWave": "FITC",
     "label": "FITC"}
  ],
  "meta": {
    "projectDescription": "",
    "datasetName": "survivin",
    "projectId": 2,
    "imageDescription": "",
    "imageTimestamp": 1277977808.0,
    "imageId": 12,
    "imageAuthor": "Will Moore",
    "imageName": "CSFV-siRNAi02_R3D_D3D.dv",
    "datasetDescription": "",
    "projectName": "siRNAi",
    "datasetId": 3
  },
  "id": 12,
  "pixel_size": {"y": 0.0663, "x": 0.0663, "z": 0.2},
  "size": {
    "width": 480,
    "c": 4,
    "z": 31,
    "t": 1,
    "height": 480
  },
  "tiles": false
}
```

For large tiled images, the following data is also included:

```
{
  "tiles": true,
}
```
"tile_size": {  
    width: 256,
    height: 256
  },
"levels": 5,
"zoomLevelScaling": {  
  0: 1,
  1: 0.25,
  2: 0.0625,
  3: 0.0312,
  4: 0.0150
},
}

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

14.7 Embedding an OMERO.web viewport in a web page

Note: These examples are intended to be used with images that have been added to the PUBLIC group with a Public member in OMERO, making them publicly available. To see how to configure public URL filters, see the Public data in the repository section.

14.7.1 OMERO.web viewer in iframe

Insert the following:

```html
<div id="omeroviewport"><iframe width="850" height="600" src="http://localhost:8000/webclient/img_detail/IMAGE_ID/" id="omeroviewport" name="omeroviewport"></iframe></div>
```

14.7.2 Launching OMERO.web viewer

Use the following code to reference the scripts.

```html
<script type="text/javascript">
    function openPopup(url) {
        owindow = window.open(url, 'anew', config='height=600,width=850,left=50,top=50,toolbar=no,menubar=no,scrollbars=yes,resizable=yes,location=no,directories=no,status=no');
        if(!owindow.closed) owindow.focus();
        return false;
    }
</script>
```

Then in `<BODY>` insert the following:

```html
<a href="#" onclick="return openPopup('http://localhost:8000/webclient/img_detail/IMAGE_ID/')">Open viewer</a>
```

14.7.3 Customizing the content of the embedded OMERO.web viewport

This section demonstrates how to embed an OMERO.web image viewer in any HTML page, allowing use of resources directly from an OMERO server.
Provided the image corresponding to IMAGE_ID is in the PUBLIC group, it can be accessed via the link: http://your_host/webgateway/img_detail/IMAGE_ID/. Please remember that public images must be in a public group where a public user can access them.

Use the following code to load stylesheets and scripts.

```html
<link rel="stylesheet" type="text/css" href="http://your_host/static/omeroweb.viewer.min.css">
<script type="text/javascript" src="http://your_host/static/omeroweb.viewer.min.js"></script>
```

Then create the small JavaScript with associated stylesheet which allows you to view particular image defined by IMAGE_ID.

```html
<style type="text/css">
 .viewport {
   height: 500px;
   width: 800px;
   padding: 10px;
 }
</style>

<script type="text/javascript">
$(document).ready(function () {

   /* Prepare the viewport */
   viewport = $.WeblitzViewport($("#viewport"), "http://your_host/webgateway/", {
    'mediaroot': "http://your_host/static/"
   });

   /* Load the selected image into the viewport */
   viewport.load(IMAGE_ID);

});
</script>
```

Then in `<BODY>` insert the following:

```html
<div id="viewport" class="viewport"></div>
```

The viewport can be made more interactive by adding buttons or links to allow display of scalebars, ROIs, zooming and selection of channels. Full examples of how to embed microscopy or Whole Slide Image are available in the Webtest Github repository.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 14.8 Writing OMERO.web views

This page contains info on how to write your own views.py code, including documentation on the `webclient/views.py` and `webgateway/views.py` code. Although we aim to provide some useful notes and examples here, you will find the best source of examples is the code itself.

---

24[https://github.com/openmicroscopy/webtest/tree/master/webtest/templates/webtest/examples](https://github.com/openmicroscopy/webtest/tree/master/webtest/templates/webtest/examples)

14.8.1 @Decorators

Decorators in Python are functions that ‘wrap’ other functions to provide additional functionality. They are added above a method using the @ notation. We use them in the OMERO.web framework to handle common tasks such as login (getting connection to OMERO server) etc.

@login_required()

The login_required decorator uses parameters in the ‘request’ object to retrieve an existing connection to OMERO. In the case where the user is not logged in, they are redirected to a login page. Upon login, they will be redirected back to the page that they originally tried to view. The method that is wrapped by this decorator will be passed a ‘conn’ Blitz Gateway connection to OMERO.

Note: login_required is a class-based decorator with several methods that can be overwrittten to customize its functionality (see below). This means that the decorator MUST be instantiated when used with the @ notation, i.e.

@login_required() NOT @login_required # this will give you strange error messages

A simple example of @login_required() usage (in webtest/views.py). Note the Blitz Gateway connection “conn” retrieved by @login_required() is passed to the function via the optional parameter conn=None.

from omeroweb.decorators import login_required

@login_required()

def dataset(request, datasetId, conn=None, **kwargs):
    ds = conn.getObject("Dataset", datasetId)
    return render(request, 'webtest/dataset.html', {'dataset': ds})

or

from omeroweb.decorators import login_required, render_response

@login_required()
@render_response()

def dataset(request, datasetId, conn=None, **kwargs):
    ds = conn.getObject("Dataset", datasetId)
    context['template'] = 'webtest/dataset.html'
    context['dataset'] = ds
    return context

login_required logic

The login_required decorator has some complex connection handling code, to retrieve or create connections to OMERO. Although it is not necessary to study the code itself, you may find it useful to understand the logic that is used (see Flow Diagram). As mentioned above, we start with a HTTP request (top left) and either a connection is returned (bottom left) OR we are redirected to login page (right).

Note: Options to configure a “public user” are described in the Public data in the repository documentation.

Extending login_required

The base login_required class can be found in omeroweb/decorators.py. It has a number of methods that can be overwrittten to customize or extend its functionality. Again, it is best to look at an example of this. See webclient/decorators.py

26https://github.com/openmicroscopy/webtest/blob/master/webtest/views.py
to see how the base omeroweb.decorators.login_required has been extended to configure the conn connection upon login, handle login failure differently etc.

render_response

This decorator handles the rendering of view methods to HttpResponse. It expects that wrapped view methods return a dict. This allows:

- the template to be specified in the method arguments OR within the view method itself
- the dict to be returned as json if required
- the request is passed to the template context, as required by some tags etc
- a hook is provided for adding additional data to the context, from the Blitz Gateway documentation or from the request.

Note: Using `@render_response` guarantees using special `RequestContext` class\(^{27}\) which always uses `django.template.context_processors.csrf` (no matter what template context processors are configured in the TEMPLATES setting). For more details see `CSRF`.

Extending render_response

The base render_response class can be found in omeroweb/decorators.py. It has a number of methods that can be overwritten to customize or extend its functionality. Again, it is best to look at an example of this. See `webclient/decorators.py` to see how the base omeroweb.decorators.render_response has been extended to configure HttpResponse and its subclasses.

14.8.2 Style guides

Tips on good practice in `views.py` methods and their corresponding URLs.

- Include any required arguments in the function parameter list. Although many `views.py` methods use the `kwargs` parameter to accept additional arguments, it is best not to use this for arguments that are absolutely required by the method.
- Specify default parameters where possible. This makes it easier to reuse the method in other ways.
- Use keyword arguments in URL regular expressions. This makes them less brittle to changes in parameter ordering in the views.
- Similarly, use keyword arguments for URLs in templates

\(^{27}\)https://docs.djangoproject.com/en/1.8/ref/templates/api/#subclassing-context-requestcontext
14.8.3 OMERO.web error handling

Django comes with some nice error handling functionality. We have customized this and also provided some client-side error handling in JavaScript to deal with errors in AJAX requests. This JavaScript can be included in all pages that require this functionality. Errors are handled as follows:

- **404** - simply displays a 404 message to the user
- **403** - this is ‘permission denied’ which probably means the user needs to login to the server (e.g. session may have timed out). The page is refreshed which will redirect the user to login page.
- **500** - server error. We display a feedback form for the user to submit details of the error to our QA system - POSTs to “qa.openmicroscopy.org.uk:80”. This URL is configurable in settings.py.

In general, you should not have to write your own error handling code in views.py or client side. The default behavior is as follows:

**With Debug: True (during development)**

Django will return an HTML page describing the error, with various parameters, stack trace etc. If the request was AJAX, and you have our JavaScript code on your page then the error will be handled as described (see above). NB: with Debug True, 500 errors will be returned as HTML pages by Django but these will not be rendered as HTML in our feedback form. You can use developer tools on your browser (e.g. Firebug on Firefox) to see various errors and open the request in a new tab to display the full debug info as HTML.

**With Debug: False (in production)**

Django will use its internal error handling to produce standard 404, 500 error pages. We have customized this behavior to display our own error pages. The 500 error page allows you to submit the error as feedback to our QA system. If the request is AJAX, we return the stack trace is displayed in a dialog which also allows the error to be submitted to QA.

**Custom error handling**

If you want to handle certain exceptions in particular ways you should use appropriate try/except statements.

This is only advised for trivial errors, where you can give the user a simple message, e.g. “No Objects selected, please try again”, or if the error is well understood and you can recover from the error in a reasonable way.

For ‘unexpected’ server errors, it is best to allow the exception to be handled by Django since this will provide a lot more info to the user (request details etc.) and format HTML (both with Debug True or False).

If you still want to handle the exception yourself, you can provide stack trace alongside a message for the user. If the request is AJAX, do not return HTML, since the response text will be displayed in a dialog box for the user (not rendered as HTML).

```python
try:
    # something bad happens
except:
    # log the stack trace
    logger.error(traceback.format_exc())
    # message AND stack trace
```
err_msg = "Something bad happened! \n \n%s" % traceback.format_exc()
if request.is_ajax():
    return HttpResponseServerError(err_msg)
else:
    # render err_msg with a custom template
    return HttpResponseServerError(content)

---

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 14.9 Writing page templates in OMERO.web

This page documents the various base templates that are used by the webclient and describes how to extend these to create your own pages with the OMERO.web look and feel.

You can use these templates in a number of ways, but there are 2 general scenarios that are detailed below:
- You want a page header to look like the webclient, but you do not need any data or connection to an OMERO server.
- You want a page that looks and behaves like it is part of the webclient application, including data from the OMERO server.

#### 14.9.1 Django templates

We use Django templates for the OMERO.web pages. See docs here: [templates](https://docs.djangoproject.com/en/1.8/ref/templates/). We have designed a number of OMERO.web base templates that you can extend. The base templates live in the `webgateway` app under `omeroweb/webgateway/templates/webgateway/base`. You can use these to make pages that do not require an OMERO login (e.g. public home page) etc.

If you want your pages to extend the webclient application, you can use templates from `omeroweb/webclient/templates/webclient/base`.

These templates are described in more detail below.

---

Figure 14.3: The `base_header.html` template extended in webtest with dummy content

You can use these templates in a number of ways, but there are 2 general scenarios that are detailed below:
- You want a page header to look like the webclient, but you do not need any data or connection to an OMERO server.
- You want a page that looks and behaves like it is part of the webclient application, including data from the OMERO server.

#### 14.9.1 Django templates

We use Django templates for the OMERO.web pages. See docs here: [templates](https://docs.djangoproject.com/en/1.8/ref/templates/). We have designed a number of OMERO.web base templates that you can extend. The base templates live in the `webgateway` app under `omeroweb/webgateway/templates/webgateway/base`. You can use these to make pages that do not require an OMERO login (e.g. public home page) etc.

If you want your pages to extend the webclient application, you can use templates from `omeroweb/webclient/templates/webclient/base`.

These templates are described in more detail below.

---

Figure 14.3: The `base_header.html` template extended in webtest with dummy content

You can use these templates in a number of ways, but there are 2 general scenarios that are detailed below:
- You want a page header to look like the webclient, but you do not need any data or connection to an OMERO server.
- You want a page that looks and behaves like it is part of the webclient application, including data from the OMERO server.

#### 14.9.1 Django templates

We use Django templates for the OMERO.web pages. See docs here: [templates](https://docs.djangoproject.com/en/1.8/ref/templates/). We have designed a number of OMERO.web base templates that you can extend. The base templates live in the `webgateway` app under `omeroweb/webgateway/templates/webgateway/base`. You can use these to make pages that do not require an OMERO login (e.g. public home page) etc.

If you want your pages to extend the webclient application, you can use templates from `omeroweb/webclient/templates/webclient/base`.

These templates are described in more detail below.

---

Figure 14.3: The `base_header.html` template extended in webtest with dummy content

You can use these templates in a number of ways, but there are 2 general scenarios that are detailed below:
- You want a page header to look like the webclient, but you do not need any data or connection to an OMERO server.
- You want a page that looks and behaves like it is part of the webclient application, including data from the OMERO server.

#### 14.9.1 Django templates

We use Django templates for the OMERO.web pages. See docs here: [templates](https://docs.djangoproject.com/en/1.8/ref/templates/). We have designed a number of OMERO.web base templates that you can extend. The base templates live in the `webgateway` app under `omeroweb/webgateway/templates/webgateway/base`. You can use these to make pages that do not require an OMERO login (e.g. public home page) etc.

If you want your pages to extend the webclient application, you can use templates from `omeroweb/webclient/templates/webclient/base`.

These templates are described in more detail below.

---

Figure 14.3: The `base_header.html` template extended in webtest with dummy content

You can use these templates in a number of ways, but there are 2 general scenarios that are detailed below:
- You want a page header to look like the webclient, but you do not need any data or connection to an OMERO server.
- You want a page that looks and behaves like it is part of the webclient application, including data from the OMERO server.

#### 14.9.1 Django templates

We use Django templates for the OMERO.web pages. See docs here: [templates](https://docs.djangoproject.com/en/1.8/ref/templates/). We have designed a number of OMERO.web base templates that you can extend. The base templates live in the `webgateway` app under `omeroweb/webgateway/templates/webgateway/base`. You can use these to make pages that do not require an OMERO login (e.g. public home page) etc.

If you want your pages to extend the webclient application, you can use templates from `omeroweb/webclient/templates/webclient/base`.

These templates are described in more detail below.
14.9.2 Getting Started

Within your OMERO.web app, create a new page template and add this line at the top:

{% extends "webgateway/base/base_header.html" %}

Now add the page content in a ‘content’ block like this:

{% block content %}
    Your page content goes here
{% endblock %}

You can now save this template and view the page. It should look something like the screen-shot above. You could add a ‘title’ block to set the page <title>

{% block title %}
    My OMERO.web app page
{% endblock %}

Additional blocks can be used to customize the page further. See below for more details.

14.9.3 Using Webclient templates

Webclient templates can be used in exactly the same way, for example try using this at the top of the page you created above:

{% extends "webclient/base/base_container.html" %}

However, this template will need various pieces of data to be in the page context that Django uses to render the page. You will need to use the @login_required() and @render_response() decorators on your views.py methods in order to retrieve this info and pass it to the template. See Writing OMERO.web views for more details.

If you have used the ‘content’ block on this page (as described above) you will see that your page content fills the whole area under the header. However, if you want to use the same 3 column layout as the webclient, you can replace your ‘content’ block with:

{% block left %}
    Left column content
{% endblock %}

{% block center %}
    Center content
{% endblock %}

{% block right %}
    Right column content
{% endblock %}

This should give you something like the screen-shot below.

14.9.4 Extending templates

You should aim to create a small number of your own base templates, extending the OMERO.web webgateway or webclient templates as required. If you extend all of your own pages from a small number of your own base templates, then you will find it easier to change things in future. For example, any changes in our ‘webgateway’ templates will only require you to edit your own base templates.

Here is a full list of the templates under omeroweb/webgateway/templates/webgateway/base with more details below:
Figure 14.4: The `webclient/base/base_container.html` template extended in webtest with dummy content

- **base_html.html** - This provides the base `<html>` template with blocks for ‘link’ (for CSS) ‘title’ ‘script’ and ‘body’. It is extended by every other template. Usage: `{% extends "webgateway/base/base_html.html" %}

- **base_frame.html** - This adds jQuery and jQuery-ui libraries to a blank page. Used for popup windows etc. Usage: `{% extends "webgateway/base/base_frame.html" %}

- **base_header.html** - This also extends base_html.html adding all the header and footer components that are used by the webclient. See screen-shot above. More details below.

- **base_main.html** - This adds jQuery and jQuery-ui libraries to the base_header.html template. Used for popup windows etc. Usage: `{% extends "webgateway/base/base_main.html" %}

- **container2.html, container3.html** - These templates extend the base_header.html template, adding a 2 or 3 column layout in the main body of the page. container3.html is used by the webclient for the base_container example above.

### Webtest examples

You can find examples of how to extend the base templates in the `webtest` application within the `webtest/templates/webtest/webgateway` directory. If you install the webtest app, you can view the template examples live at `<your-server-name>/webtest/webgateway_templates/base_header/>`

The link is to an example that extends base_header.html and contains links to all the other webtest examples. These pages indicate the names of the various template “blocks” that have been used to add content to different parts of the page (also see below for block names).

#### 14.9.5 Content blocks

These blocks can be used to add content to specific points in the page.

**Note:** It is important to consider using `{{ block.super }}` if you want to include the content from the parent template. This is critical for the “link” and “script” blocks, which are used to add `<link>` and `<script>` elements to the head of the page. If you forget to use `{% if block.super %}` then you will remove all the CSS and JavaScript links required by the parent template.

See `base_header.html` for full template details.

- **link:** used to add CSS with `<link>` blocks to the page head e.g.
• script - used to add JavaScript with <script> blocks to the page head
• title - add text here for the page <title>.
• head - another block for any extra head elements
• middle_header_right - add content to the right of the main header
• middle_header_left - add content to the left of the main header
• content - main page content.

container2.html, container3.html

These templates have all the same blocks as base_header.html since they extent it (see above). In addition, they also add the following blocks:

• left: The left column (NOT in container2.html)
• center: The middle column
• right: The right column

See container3.html for full template details.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

14.10 Cross Site Request Forgery protection

CSRF is an attack which forces an end user to execute unwanted actions on a web application in which they are currently authenticated. For more details see Cross-Site Request Forgery.

OMERO.web provides easy-to-use protection against Cross Site Request Forgeries, for more information see Django documentation.

The first defense against CSRF attacks is to ensure that GET requests (and other ‘safe’ methods, as defined by 9.1.1 Safe Methods, HTTP 1.1, RFC 2616) are only reading data from the server.

Requests that write data to the server should only use methods such as POST, PUT and DELETE. These requests can then be protected as follows:

• By default OMERO.web has the middleware django.middleware.csrf.CsrfViewMiddleware added to the list of middleware classes.

• In any template that uses a POST form, use the csrf_token tag inside the <form> element if the form is for an internal URL, e.g.:

\[
\text{<form action="." method="post">{% csrf_token %}}
\]

33https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)
34https://docs.djangoproject.com/en/1.8/ref/csrf/
Note: This should not be done for POST forms that target external URLs, since that would cause the CSRF token to be leaked, leading to a vulnerability.

• On each XMLHttpRequest set a custom X-CSRFToken header to the value of the CSRF token and pass the CSRF token in data with every AJAX POST request. If your custom template already benefits from loading built-in jQuery template you do not need to do anything as it already loads webgateway/js/ome.csrf.js. Otherwise simply import the script as follows:

  <script type="text/javascript" src="{% static "webgateway/js/ome.csrf.js" %}" /></script>

For more details see CSRF for ajax36.

The Django framework also offers decorator methods that can help you protect your view methods and restrict access to views based on the request method. For more details see Django decorators37.

By default OMERO.web provides a built-in view that handles all unsafe incoming requests failing with 403 Forbidden response if the CSRF token has not been included with a POST form.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 14.11 The OMERO.web client application

The webclient Django app provides the main OMERO.web client UI.

Data is retrieved from OMERO using the [OMERO Python language bindings](https://docs.ome.org/) and is rendered to HTML using [Django templates](https://docs.djangoproject.com/en/1.8/topics/templates/) before being sent to the browser. Additional javascript (jQuery-based) functionality in the browser is used to update the UI in response to user interactions such as browsing or modifying the data. This will often involve AJAX calls to load more data as HTML or JSON objects.

Note: The webclient should NOT be considered as a stable public API. URLs and methods within the app are purely designed to provide internal functionality for the webclient itself and may change in minor releases without warning.

#### 14.11.1 Top level pages

There are a small number of top level HTML pages that the user will start at. These are all handled by the load_template view:

• / (homepage delegates to /userdata)
• /userdata
• /usertags
• /public
• /history
• /search

These pages contain many different jQuery scripts that run when the page loads, to setup event listeners and to load additional data.

Additional top-level pages are used in popup windows for running scripts and downloading data.

---

36https://docs.djangoproject.com/en/1.8/ref/csrf/#ajax
37https://docs.djangoproject.com/en/1.8/topics/http/decorators/
38https://docs.ome.org/
14.11.2 JsTree

A jsTree is used by the userdata, usertags and public pages to browse hierarchical data. Each time a node is expanded, it uses appropriate AJAX calls to load children as json data. Further POST or DELETE AJAX calls are made to modify the data hierarchy by creating or deleting links between objects.

Selection changes in the jsTree and centre panel thumbnails cause events to be triggered by jQuery on the $("body") element of the page, allowing other scripts to listen for these events. These are used to load selected data into the centre and right panels. The HTML for these panels contains additional scripts that also run when they load to setup their own event listeners.

There is also a global `update_thumbnails_panel` function that can be called by any script that needs to refresh the centre panel. For example, when the jsTree is used to add or remove Images from a selected Dataset.

14.11.3 Switching Groups and Users

The current group and user are stored in the HTTP session as `request.session['active_group']` and `request.session['user_id']`. These are used to define the data that is loaded in the main userdata and usertags pages.

The group and user are switched by the Groups and Users menu, which updates the session and reloads the page.

14.11.4 Show queries

Data in OMERO can be linked from the webclient with URLs of the form `/webclient/?show=image-23|image-34`. In the load_template view, the first object is queried from OMERO and its parent containers are also loaded. The owner and group of the top container is used to set the `active_group` and `user_id` so that the main page loads the appropriate data hierarchy.

The jsTree does its own lookup from the query string, retrieving json data and using this to expand the appropriate nodes to show the specified objects.

14.11.5 Javascript code

The majority of javascript code is jQuery-based code that is embedded within the HTML templates and is run when the page loads.

Additional code is in static scripts, with functions generally name-spaced under an OME module.

---

39 https://docs.djangoproject.com/en/1.8/topics/http/sessions/
15.1 Architecture

15.1.1 Logical view

OMERO.insight is logically organized in two layers

The Agents layer contains the logic to manage user interaction. It contains coarse grained components which we call agents, that are each responsible for a specific aspect of the application’s functionality:

- The Data Manager provides the user with the GUI functionality to access their data, metadata and visualize large image sets.
- The Viewer is a tool to visualize and tune 5D images.
- The Measurement Tool is a tool to perform basic measurement.

Note: If you want to add a new agent, go to How to build an agent.

These agents are internally organized according to the MVC (Model-View-Controller\(^1\)) pattern, PAC (Presentation-Abstraction-Control\(^2\)) pattern, or a combination of the two. They rely on the services provided by the bottom layer, the Container, to accomplish their tasks.

The Container layer manages the agents life-cycle and provides them with services to:

- Communicate without having to know each other (Event bus).
- Access the OMERO Server (data management and image services).
- Transform entries in configuration files into objects and then access them (Configuration).
- Log messages (log service) and notify the user (user notification service) of runtime errors.
- Cache data (cache service).
- Provide a common top level window to plug their GUI’s (Taskbar).

15.1.2 Initialization of Agents

Agents let the container create them and then manage their life-cycle. This is achieved through the use of a common interface, Agent, that all agents have to implement and by requiring every agent to have a public no-arguments constructor. The Agent interface plays the role of a Separated Interface (Fow\(^3\)), decoupling the container from knowledge of concrete agents. This way, new agents can be plugged in.

---

\(^1\) http://en.wikipedia.org/wiki/Model-view-controller

\(^2\) http://en.wikipedia.org/wiki/Presentation-abstraction-control

\(^3\) http://martinfowler.com/books
At start-up the container finds out which are the agents’ implementation classes from its configuration file, instantiates every agent by reflection (using the no-arguments constructor) and then reads each agent’s configuration file (Fow⁴). The configuration entries in this file are turned into objects and collected into a map-like object, which is then passed to the agent. This map object also contains pointers to the container’s services. We can think of this object as a Registry (Fow⁵).

⁴http://martinfowler.com/books
⁵http://martinfowler.com/books
There is one Registry containing pointers to the container’s services for each agent, so configuration entries are private to each agent - container’s services are shared among all agents though. Agents access the Registry object through the Registry interface. The life-cycle of an agent is as follow:

![Figure 15.2: OMERO.insight agent lifecycle](image)

15.1.3 Interaction among Agents

Interactions among agents are event-driven. Agents communicate by using a shared *Event bus* provided by the container. The event bus is an event propagation mechanism loosely based on the Publisher-Subscriber\(^6\) pattern and can be regarded as a time-ordered event queue - if event A is posted on the bus before event B, then event A is also delivered before event B.

15.1.4 Process view

All agents run synchronously within the *Swing* dispatching thread. All container’s services are called within Swing event handlers and thus run within the Swing dispatching thread. To see how to retrieve data from an OMERO server, go to the [Retrieve data from server](#) page.

See also:

*Organization*, *Event bus*

---

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

15.2 Configuration

The container provides a flexible and extensible configuration mechanism. Each agent has its own configuration file which is parsed at start-up by the container. The configuration entries in this file are turned into objects and collected into a map-like object, which is then passed to the agent. This map object also contains pointers to the container’s services. Thus, we can think of

---

\(^6\)http://en.wikipedia.org/wiki/Publish%2Fsubscribe
this object as a Registry. There is one Registry for each agent, so configuration entries are private to each agent - container’s services are shared among all agents though. The container also has its own configuration file and Registry.

The container maintains a set of predefined bindings that are used to convert a configuration entry into an object - such as a String, Integer, Font, IconFactory, etc. However, agents can specify custom handlers for converting a configuration entry into an object.

15.2.1 Structure

Configuration files are XML files which declare only two elements:

```xml
<element name="entry" minOccurs="0" maxOccurs="unbounded">
  <complexType>
    <simpleContent>
      <extension base="string">
        <attribute name="name" type="string" use="required"/>
        <attribute name="type" type="string" default="string"/>
        <simpleType>
          <restriction base="string">
            <enumeration value="string"/>
            <enumeration value="integer"/>
            <enumeration value="float"/>
            <enumeration value="double"/>
            <enumeration value="boolean"/>
          </restriction>
        </simpleType>
      </extension>
    </simpleContent>
  </complexType>
</element>

<element name="structuredEntry" minOccurs="0" maxOccurs="unbounded">
  <complexType>
    <sequence>
      <any maxOccurs="unbounded"/>
    </sequence>
    <attribute name="name" type="string" use="required"/>
    <attribute name="type" type="string" default="map"/>
  </complexType>
</element>
```

The `entry` and `structuredEntry` elements are used to specify name-value pairs in the actual configuration files. Both elements have a required name attribute whose content is used as a key for accessing the entry value from within the application. In the case of the entry element, the value is a string – this element can thus be used for “classic” name-value style, such as:

```xml
<entry name="/some/name" type="boolean">true</entry>
```

On the other hand, the value of `structuredEntry` can be made up by any arbitrary sequence of tags.

In both cases (`entry` and `structuredEntry`), the entry value is returned to the application as an object and the type attribute dictates how the entry value is turned into an object. Only “string”, “integer”, “float”, “double” and “boolean” may be used for the type attribute within the entry element, whose content is parsed into a Java String, Integer, Float, Double or Boolean object according to the content of the type attribute – if this attribute is missing, then “string” is assumed. For example, say you write the following into an agent’s configuration file:

```xml
<entry name="/some/name" type="boolean">true</entry>
```

This will make a Boolean object (set to hold true) available to the agent – the key for accessing the object will be the string “/some/name”.

Things work similarly for the `structuredEntry` element. In this case, the content of the type attribute can be specified to be the fully qualified name of the class that will handle the transformation of the entry value into an object. This is provided so that agents may specify custom handlers for custom configuration entries. For example, an agent’s configuration file could contain the entry:

```xml
OMERO Documentation, Release 5.2.6

<structuredEntry name="/some/name" type="some.pkg.SomeHandler">
  <tag_1>aValue</tag_1>
  <tag_2>anotherValue</tag_2>
</structuredEntry>

In this case, an instance of some.pkg.SomeHandler will be created to transform the contents of the entry (that is tag_1 and tag_2) into a custom object. Obviously enough, the tags contained within a structuredEntry have to be exactly the tags that the handler expects.

If no type attribute is specified, then it is assumed type = "map", which results in the entry’s contents being parsed into a Map object. Each child tag is assumed to be a simple tag with a string content, like in the following example:

<structuredEntry name="/some/name">
  <key_1>value_1</key_1>
  <key_2>value_2</key_2>
</structuredEntry>

Each child tag’s name is a key in the map and the tag’s content is its value – the above would generate the map: (key_1, value_1), (key_2, value_2).

Some predefined structured entries are supplied by the container for common cases (icons and font entries) and for use by the container only (OMERO and agents entries). Here’s an excerpt from the container’s configuration file:

<container>
  <services>
    <structuredEntry name="/services/OMERODS" type="OMERODS">
      <port>1099</port>
    </structuredEntry>
  </services>
  <agents>
    <structuredEntry name="/agents" type="agents">
      <agent>
        <name>Viewer</name>
        <!-- The class tag specifies the FQN of the agent’s class. -->
        <class>org.openmiscroscopy.shoola.agents.viewer.Viewer</class>
        <!-- The config tag specifies the name of the agent's configuration file. This file has to be placed in the config directory under the installation directory. -->
        <config>viewer.xml</config>
      </agent>
      . . . a similar entry for every other agent
    </structuredEntry>
  </agents>
  <resources>
    <iconFactories>
      <!-- This entry can be used in agents’ configuration files as well. It is turned into an instance of: org.openmicroscopy.shoola.env.config.IconFactory
      This object can then be used to retrieve any image file within the directory pointed by the location tag. -->
      <structuredEntry name="/resources/icons/DefaultFactory" type="icons">
        <!-- The location tag specifies the FQN of the package that contains the icon files. -->
        <location>org.openmicroscopy.shoola.env.ui.graphx</location>
      </structuredEntry>
      . . . more similar entries
    </iconFactories>
    <fonts>
      <!-- This entry can be used in agents’ configuration files as well. It is turned into an instance of java.awt.Font. -->
      <structuredEntry name="/resources/fonts/Titles" type="font">
        <family>SansSerif</family>
      </structuredEntry>
    </fonts>
  </resources>
</container>
The configuration parser only takes the entry and structuredEntry tags into account and ignores all the others. It may be useful to group sets of related entries together, as shown above.

The classes that encompass the machinery for parsing configuration files and building registries are depicted by the following UML class diagram.

![UML class diagram](image)

**Figure 15.3: OMERO.insight configuration**

The Entry abstract class sits at the base of a hierarchy of classes that represent entries in configuration files. It represents a name-value pair, where the name is the content of the name attribute of a configuration entry (which is stored by the name field) and the value is the object representing the entry’s content. As the logic for building an object from the entry’s content depends on what is specified by the type attribute, this class declares an abstract getValue method which subclasses implement to return the desired object – we use polymorphism to avoid conditional logic. So we have subclasses (StringEntry, IntegerEntry, IconFactoryEntry, etc.) to handle the content of an entry tag (either entry or structuredEntry) in correspondence of each predefined value of the type attribute (“string”, “integer”, “icons”, and so on). Given an entry tag, the createEntryFor static method (which can be considered a Factory Method) creates a concrete Entry object to handle the conversion of that tag’s content into an object. Subclasses of Entry implement the setContent method to grab the tag’s content, which is then used for building the object returned by the implementation of getValue().

The Registry Interface declares the operations to be used to access configuration entries and container’s services.

The RegistryImpl class implements the Registry interface. It maintains a map of Entry objects, which are keyed by their name attribute and represent entries in the configuration file. It also maintains references to the container’s services into member fields – as services are accessed frequently, this ensures \( o(1) \) access time.
The **Parser** class is in charge of parsing a configuration file, extracting entries (only *entry* and *structuredEntry* tags are taken into account), obtain an **Entry** object to represent each of those entries and add these objects to a given **RegistryImpl** object.

### 15.2.2 Dynamics

How a configuration file is parsed and the corresponding Registry is built:

A **RegistryImpl** object is created with an empty map. Then a **Parser** object is created passing the path to the configuration file and the **RegistryImpl** object. At this point `parse()` is invoked on the **Parser** object. The configuration file is read (the XML parsing is handled by built-in JAXP libraries) and, for each configuration entry (that is, either *entry* or *structuredEntry* tag), `createEntryFor()` is called to obtain a concrete **Entry** object, which will handle the conversion of the tag’s content into an object. This **Entry** object is then added to the map kept by the **RegistryImpl** object.

In order to find out which class is in charge of handling a given tag, the **Entry** class maintains a map, `contentHandlers`, whose keys are the predefined values used for the type attribute (“string”, “integer”, “icons”, etc.) and values are the fully qualified names of the handler classes. Given a tag, `createEntryFor()` uses the content of the type attribute (or “string” if this attribute is missing) to look up the class name in the map and then creates an instance by reflection - all **Entry**'s subclasses are supposed to have a no-args constructor. If the class name is not found in the map, then the content of the type attribute is assumed to be a valid fully qualified name of an **Entry**'s subclass. This allows for agents to specify custom handlers – as long as the handler extends **Entry** and has a public no-args constructor.

Notice that the **RegistryImpl** object adds the couple `(e.getName(), e)` to its map. Because the **Entry** class takes care of setting the name field to the content of the name attribute within the entry tag, the application code can subsequently access `e` by specifying the value of the name attribute to `lookup()`. The above outlined process is repeated for each configuration file so that the configuration entries of each agent (and the container) are kept in separate objects – a **RegistryImpl** is created every time. Because every agent is then provided with its own **RegistryImpl** object, the configuration entries are private to each agent. However, the container configures all **RegistryImpl** objects with the same references to its services.

See also:

*Directory contents*

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event
of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

## 15.3 Contributing to OMERO.insight

### Note:
See the [Contributing Developer](http://www.openmicroscopy.org/site/support/contributing/) documentation for more information about contributing to OME projects in general.

### 15.3.1 Getting started with OMERO.insight

Getting started with OMERO.insight entails that you have an *OMERO.server* already deployed.

### 15.3.2 Installing from source

Since January 2011, the OMERO.insight code base is part of the OMERO code base. See [Installing OMERO from source](http://git.openmicroscopy.org), to check out code using http://git.openmicroscopy.org.

#### Requirements

- Install a Java 7+ Development Kit (JDK), available from [Java SE Downloads](http://www.oracle.com/technetwork/java/javase/downloads/index.html) and required for both the OMERO server and client code. Set the `JAVA_HOME` environment variable to your JDK installation.

#### Running code

It is helpful to set up the project in [Eclipse](http://www.eclipse.org). Because the OMERO Java and Python source files are encoded in UTF-8, ensure that the encoding in Eclipse (Preferences → General → Workspace → Text file encoding) is also set to UTF-8.

#### Build system

See [Build System](http://www.openmicroscopy.org/site/support/contributing//ci-omero.html) for details.

The OME project currently uses [Jenkins](http://jenkins-ci.org) (formerly known as hudson) as a continuous integration server available [here](https://ci.openmicroscopy.org/). OMERO.insight is built as part of the “OMERO” jobs. See the [contributing developer continuous integration](http://www.openmicroscopy.org/site/support/contributing//ci-omero.html) documentation for full details.

### Note:
This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

## 15.4 Directory contents

The repository of the software artifacts is organized as follows:

- **build**: This directory contains the tools to compile, run, test, and deliver the application.
- **config**: Various configuration files required by the application to run.

---

7 [http://www.openmicroscopy.org/site/support/contributing/](http://www.openmicroscopy.org/site/support/contributing/)
9 [http://www.eclipse.org](http://www.eclipse.org)
10 [http://jenkins-ci.org](http://jenkins-ci.org)
11 [https://ci.openmicroscopy.org/](https://ci.openmicroscopy.org/)
• **docgen**: Documentation artifacts that are used to build actual documents. These are organized in two sub-directories: javadoc and xdocs. The former only contains resources (like CSS files) to generate programmer’s documentation – the actual documentation contents are obtained from the source code. The latter contains both resources (like stylesheets and DHTML code) to generate all other kinds of documentation (like design and users documents) and the actual documentation contents in the form of XML/HTML files.

• **launch**: Launcher scripts and installation instructions bundled with the default application distribution file. Its sub-dirs contain further resources to build platform-specific distributions.

• **SRC**: Contains the application source files.

• **TEST**: The test code.

• **README**: The README file.

---

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 15.5 Event bus

Interactions among agents are event-driven. Agents communicate by using a shared event bus provided by the container. The event bus is an event propagation mechanism loosely based on the [Publisher-Subscriber](http://en.wikipedia.org/wiki/Publish/subscribe) pattern and can be regarded as a time-ordered event queue - if event A is posted on the bus before event B, then event A is also delivered before event B.

Events are fired by creating an instance of a subclass of `AgentEvent` and by posting it on the event bus. Agents interested in receiving notification of `AgentEvent` occurrences implement the `AgentEventListener` interface and register with the event bus. This interface has a callback method, `eventFired`, that the event bus invokes in order to dispatch an event. A listener typically registers interest only for some given events - by specifying a list of `AgentEvent` subclasses when registering with the event bus. The event bus will then take care of event de-multiplexing - an event is eventually dispatched to a listener only if the listener registered for that kind of event.

#### 15.5.1 Structure

**EventBus**

- Defines how client classes access the event bus service.
- A client object (subscriber) makes/cancels a subscription by calling `register()/remove()`.
- A client object (publisher) fires an event by calling `postEvent()`.

**AgentEvent**

- Ancestor of all classes that represent events.
- Source field is meant to be a reference to the publisher that fired the event.
- An event is "published" by adding its class to the events package within the agents package.

**AgentEventListener**

- Represents a subscriber to the event bus.
- Defines a callback method, `eventFired`, that the event bus invokes in order to dispatch an event.

**EventBusListener**

- Concrete implementation of the event bus.
- Maintains a de-multiplex table to keep track of what events have to be dispatched to which subscribers.

---

13http://en.wikipedia.org/wiki/Publish/subscribe
15.5.2 In action

- When a subscriber invokes the register or remove method, the de-multiplex table is updated accordingly and then the event bus returns to idle.
- When a publisher invokes `postEvent()`, the event bus enters into its dispatching loop and delivers the event to all subscribers in the event notification list.
- Time-ordered event queue - if event A is posted on the bus before event B, then event A is also delivered before event B.
- Dispatching loop runs within same thread that runs the agents (Swing dispatching thread).

15.6 Event

15.6.1 Structure

We devise two common categories of events:

- Events that serve as a notification of state change. Usually events posted by agent to notify other agents of a change in its internal state.

- Events that represent invocation requests and completion of asynchronous operations between agents and some services.

In the first category fall those events that an agent posts on the event bus to notify other agents of a change in its internal state. Events in the second category are meant to support asynchronous communication between agents and internal engine. The `AgentEvent` class, which represents the generic event type, is sub-classed in order to create a hierarchy that represents the above categories. Thus, on one hand we have an abstract `StateChangeEvent` class from which agents derive concrete classes to represent state change notifications. On the other hand, the `RequestEvent` and `ResponseEvent` abstract classes are sub-classed by the container in order to define, respectively, how to request the asynchronous execution of an operation and how to represent its completion. We use the
Asynchronous Completion Token pattern to dispatch processing actions in response to the completion of asynchronous operations.

**StateChangeEvent**
- Ancestor of all classes that represent state change notifications.
- Its state field can be used to carry all state-change information.

**RequestEvent**
- Abstractly represents a request to execute an asynchronous operation.
- A concrete subclass encapsulates the actual request.
- Knows how and which processing action to dispatch upon completion of the asynchronous operation.

**CompletionHandler**
- Represents a processing action.
- Allows for all processing action to be treated uniformly.

**ResponseEvent**
- Abstractly represents the completion of an asynchronous operation.
- A concrete subclass encapsulates the result of the operation, if any.
- Knows the RequestEvent object that originated it.
- Knows how to activate the de-multiplexing of a completion event to the processing action.

### 15.6.2 In action

Follow a concrete example:
//Somewhere in the Data Manager
//Request to View an image
EventRequest req = new ViewImage((ImageData) image, null)
//Request the execution of the view call.
eventBus.postEvent(req);

//Somewhere in the Viewer Agent
public void eventFired(AgentEvent e)
{
    if (e instanceof ViewImage) handleViewImage((ViewImage) e);
}

A concrete RequestEvent encapsulates a request to execute an asynchronous operation. Asynchrony involves a separation in space and time between invocation and processing of the result of an operation: we request the execution of the operation at some point in time within a given call stack (say in methodX we make a new request and we post it on the event bus). Then, at a later point in time and within another call stack (eventFired method), we receive a notification that the execution has completed and we have to handle this completion event - which mainly boils down to doing something with the result, if any, of the operation. Recall that the ResponseEvent class is used for representing a completion event and a concrete subclass carries the result of the operation, if any. After the operation has completed, a concrete ResponseEvent is put on the event bus so that the object which initially made the request (often an agent, but, in this context, we will refer to it as the initiator, which is obviously required to implement the AgentEventListener interface and register with the event bus) can be notified that execution has completed and possibly handle the result. Thus, at some point in time the initiator’s eventFired method is called passing in the response object.
Now the initiator has to find out which processing action has to be dispatched to handle the response. Moreover, the processing action often needs to know about the original invocation context - unfortunately, we cannot relinquish the original call stack (methodX is gone). The solution is to require that a response be linked to the original request and that the initiator link a request to a completion handler (which encapsulates the processing action) before posting it on the event bus (this explains the fancy arrangement of the RequestEvent, ResponseEvent and CompletionHandler).

This way de-multiplexing matters are made very easy for the initiator. Upon reception of a completion event notification, all what the initiator has to do is to ask the response object to start the de-multiplexing process - by calling the complete method. This method calls handleCompletion() on the original request, passing in the response object. In turn, handleCompletion() calls the handle method on its completion handler, passing in both the request and the response. The right processing action has been dispatched to handle the response. Also, notice that the completion handler is linked to the request in the original invocation context, which makes it possible to provide the handler with all the needed information from the invocation context. Moreover, both the original request and the corresponding response are made available to the completion handler. This is enough to provide the completion handler with a suitable execution context - all the needed information from the original call stack is now available to the processing action.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 15.7 How to build an agent

An agent is created and managed by the container. Building an agent is done in two steps:

- write the agent code
- declare the agent in the container.xml file located in the config directory

The agent intercepts the events posted on the event bus.

Note: When a new version of the software is delivered, make sure you keep the container.xml shipped with the application and add the new agent entry to it.

### 15.7.1 Writing code

The following example creates a concrete agent MyBrowserAgent:

- Create a myBrowser package in the agents package.
- Create a class MyBrowserAgent, this class MUST implement the Agent interface to be initialized and the AgentListener to interact with other agents.

```java
public class MyBrowserAgent
    implements Agent, AgentEventListener {

    /** Reference to the registry. */
    private static Registry registry;

    //no-arguments constructor required for initialization
    public MyBrowserAgent() {}

    //Follow methods required by the Agent Interface

    //No-op implementation in general
    public void activate()
    {
        //this method will be invoked during the activation by the container
    }

    //invoked before shutting down the application
```
public boolean canTerminate() { return true; }

// not yet implemented: invoked when shutting down the application
public Map<String, Set> hasDataToSave() { return null; }

// invoked while shutting down the application
public void terminate() {}

public void setContext(Registry ctx)
{
    // Must be a reference to the Agent Registry to access services.
    registry = ctx;

    // register the events the agent listens to e.g. BrowseImage
    EventBus bus = registry.getEventBus();
    bus.register(this, BrowseImage.class);
}

// Follow methods required by the AgentEventListener Interface
public void eventFired(AgentEvent e)
{
    if (e instanceof BrowseImage) {
        // Do something
        browseImage((BrowseImage) e);
    }
}

Where to create the BrowseImage event

- Create a myBrowser package in the agents.events package.
- Create a BrowseImage event in the myBrowser package.

public class BrowseImage extends RequestEvent
{
    /** The id of the image to browse. */
    private long imageID;

    /**
     * Creates a new instance.
     * @param imageID The id of image to view.
     */
    public BrowseImage(long imageID)
    {
        if (imageID < 0)
            throw new IllegalArgumentException("ImageID not valid.");
        this.imageID = imageID;
    }

    /**
     * Returns the ID of the image to browse.
     */
    @return See above.
Listening to the BrowseImage event

To listen to events posted on the event bus, the agent MUST implement the AgentListener Interface and register the events to listen to.

- Register BrowseImage in the setContext(Registry) method of the Agent interface.
- Listen to BrowseImage in the eventFired(AgentEvent) method of the AgentListener interface.

For example, when clicking on an image in the Data Manager, the following event is posted:

```java
EventBus bus = registry.getEventBus();
bush.post(new BrowseImage(imageID));
```

The MyBrowserAgent handles the event

```java
public void eventFired(AgentEvent e) {
    if (e instanceof BrowseImage) {
        //Do something
        browseImage((BrowseImage) e);
    }
}
```

Creating an agent's view

See How to build an agent's view

15.7.2 Declaring the agent

The MyBrowserAgent needs to be declared in the container.xml.

- Open the container.xml located in the config folder (see Directory contents).
- Add the following:

```xml
<agents>
    <structuredEntry name="/agents" type="agents">
        <!-- NOTE FOR DEVELOPERS
        Add an agent tag for each of your Agents.
        The name tag specifies the human-readable name of the Agent.
        The active tag specifies if the agent is turned on or off.
        Set to true to turn the agent on, false otherwise.
        The class tag specifies the FQN of the Agent class.
        The config tag specifies the name of the Agent's configuration file within the config directory.
        -->
        <agent>
            <name>My Browser</name>
            <active>true</active>
            <class>org.openmicroscopy.shoola.agents.mybrowser.MyBrowserAgent</class>
        </agent>
    </structuredEntry>
</agents>
```
• Create a mybrowser.xml and add it to the config directory:

```xml
<?xml version="1.0" encoding="utf-8"?>
<agent name="My Browser">
  <resources>
    <iconFactories>
      <!-- This entry is turned into an instance of:
      org.openmicroscopy.shoola.env.config.IconFactory
      This object can then be used to retrieve any image file within
      the directory pointed by the location tag. -->
      <structuredEntry name="/resources/icons/Factory" type = "icons">
        <!-- The location tag specifies the FQN of the package that contains
         the icon files. -->
        <location>org.openmicroscopy.shoola.agents.myBrowser.graphx</location>
      </structuredEntry>
    </iconFactories>
    <fonts>
      <!-- This entry is turned into an instance of java.awt.Font. -->
      <structuredEntry name="/resources/fonts/Titles" type="font">
        <family>SansSerif</family>
        <size>12</size>
        <style>bold</style>
      </structuredEntry>
    </fonts>
  </resources>
</agent>
```

The file mybrowser.xml allows the agent to define specific parameters.

See also:

Organization, Retrieve data from server

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 15.8 How to build an agent’s view

This section explains how a view of the agent is created. All our agents follow the same approach. To see the code while reading the notes, go to `components/insight/SRC/org/openmicroscopy/shoola/agents/treeviewer/view`.

Using the previous example MyBrowserAgent (see How to build an agent):

1. Create a view package in the mybrowser package.
2. Create the following classes MyBrowser (interface), MyBrowserComponent, MyBrowserModel, MyBrowserControl, and MyBrowserUI. If you browse the source code, you will notice that we usually have a class used as a toolbar and a class used as a status bar. Both classes are initialized by the BrowserUI. For clarity, they have been omitted in the following diagram.
3. Create a MyBrowserFactory. This class keeps track of the MyBrowser instances created and not yet discarded. A component is only created if none of the tracked ones is displaying the data, otherwise the existing component is recycled.

14https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/insight/SRC/org/openmicroscopy/shoola/agents/treeviewer/view
15.8.1 Typical life-cycle of an agent view

The object is first created using the MyBrowserFactory

//Somewhere in the MyBrowserFactory code
private static final MyBrowserFactory singleton = new MyBrowserFactory();

public static MyBrowser getViewer(Set<ImageData> images)
{
    MyBrowserModel model = new MyBrowserModel(images);
    return singleton.getViewer(model);
}

private MyBrowser getViewer(MyBrowserModel model)
{
    Iterator v = viewers.iterator();
    MyBrowserComponent comp;
    while (v.hasNext()) {
        comp = (MyBrowserComponent) v.next();
        if (model.isSameDisplay(comp.getModel())) {
            comp.refresh(); //refresh the view.
            return comp;
        }
    }
    comp = new MyBrowserComponent(model);
    comp.initialize();
    comp.addChangeListener(this);
    viewers.add(comp);
    return comp;
}

After creation, the object is in the MyBrowser#NEW state and is waiting for the MyBrowser#activate() method to be called. Such a call usually triggers loading of the objects on the server. The object is now in the MyBrowser#LOADING state. After all the data have been retrieved, the object is in the MyBrowser#READY state and the data display is built and set on screen.

When the user quits the window, the MyBrowser#discard() method is invoked and the object transitions to the MyBrowser#DISCARDED state. At which point, all clients should de-reference the component to allow for garbage collection.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

15.9 Retrieve data from server

To retrieve data stored in an OMERO server, Agents can either:

- directly access a Data service (container service) through their Registry (in which case, the call happens in the Swing dispatching thread, so it is not possible to give user feedback by showing a progress bar for example):
OMeroDataService service = registry.getDataService();
service.getServerName()

- or retrieve data asynchronously using a Data Services View
15.9.1 Data services view

Usage

A data services view is a logical grouping of data and operations that serve a specific purpose, for example to support dataset browsing by providing easy access to datasets, thumbnails, tags, etc. A data services view is defined by an interface that extends `DataServiceView` and consists of a collection of asynchronous calls that operate on (possibly) large portions of a data model in the background.

Agents obtain a reference to a given view through their registry by specifying the view’s defining interface as follows (note the `required` cast on the returned reference):

```java
XxxView view = (XxxView) registry.getDataServicesView(XxxView.class);
```

`XxxView` is obviously a made up name for one of the sub-interfaces of `DataServiceView` contained in this package. All calls are carried out asynchronously with respect to the caller’s thread and return a `CallHandle` object which can be used to cancel execution. This object is then typically linked to a button so that the user can cancel the task, like in the following example:

```java
final CallHandle handle = view.loadSomeDataInTheBg(observer);
//The above call returns immediately, so we don't have to wait.
//While the task is carried out, we allow the user to change
//her mind and cancel the task:

cancelButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        handle.cancel();
    }
});
```

The `observer` argument to the above call is an instance of `AgentEventListener` (in `env.event`). Normally all calls within a view allow to specify this argument, which is used to provide the caller with feedback on the progress of the task and with its eventual outcome.

Specifically, as the computation proceeds in the background, `DSCallFeedbackEvents` (in `env.data.events`) are delivered to the `observer`. These event objects have a status field which contains a textual description of the activity currently being carried out within the computation and a progress indicator which is set to the percentage of the work done so far. So the indicator will be 0 for the first feedback event and, if the computation runs to completion, 100 for the last feedback event, which will always have its status field set to `null`—note that a `null` status is also possible for the previous events if no description was available at the time the event was fired. Moreover, any partial result that the computation makes available will be packed into the feedback event.

It is important to keep in mind that the computation may not run to completion—either because of an exception within the computation or because the agent cancels execution—`CallHandle.cancel()` (in `env.data.views`). In both cases, the feedback notification will not run to completion either. However, in any case a final `DSCallOutcomeEvent` (in `env.data.events`) is delivered to the `observer` to notify of the computation outcome—the event’s methods can be used to find out the actual outcome and retrieve any result or exception. Every call documents what is the returned object and what are the possible exceptions so that the caller can later cast the returned value or exception as appropriate.

Here is the code for a prototypical `observer`:

```java
public void eventFired(AgentEvent ae) {
    if (AE instanceof DSCallFeedbackEvent) { //Progress notification.
        update((DSCallFeedbackEvent) AE); //Inform the user.
    } else { //Outcome notification.
        DSCallOutcomeEvent oe = (DSCallOutcomeEvent) AE;
        switch (oe.getState()) {
            case DSCallOutcomeEvent.CANCELLED: //The user cancelled.
                handleCancellation();
                break;
```
case DSCallOutcomeEvent.ERROR:  //The call threw an exception.
    handleException(oe.getException());
    break;
  case DSCallOutcomeEvent.NO_RESULT:  //The call returned no value.
    handleNullResult();
    break;
  case DSCallOutcomeEvent.HAS_RESULT:  //The call returned a value.
    handleResult(oe.getResult());
  }
  }
  }

Because the logic is likely to be common to most of the observers, the DSCallAdapter (in env.data.events) class factors it out to provide a more convenient way to write observers. Back to our previous example, the observer could look something like the following:

```java
observer = new DSCallAdapter() {
    public void update(DSCallFeedbackEvent fe) { //Received some feedback.
        String status = fe.getStatus();
        int percDone = fe.getPercentDone();
        if (status == null)
            status = (percDone == 100) ? "Done" : ""; //Else
            "Description was not available.
        statusBar.setText(status); //A JLabel object part of the UI.
        progressBar.setValue(percDone); //A JProgressBar object part of the UI.
    }
    public void onEnd() { //Called right before any of the handleXXX methods.
        progressBar.setVisible(false); //Because the computation has finished.
    }
    public void handleResult(Object result) { //Computation returned a result.
        //We have a non-null return value. Cast it to what
        //loadSomeDataInTheBg() declared to return.
        SomeData data = (SomeData) result;
        //Update model, UI, etc.
    }
    public void handleCancellation() { //Computation was cancelled.
        UserNotifier un = registry.getUserNotifier();
        un.notifyInfo("Data Loading", "SomeData task cancelled.");
    }
    public void handleException(Throwable exc) { //An error occurred.
        UserNotifier UN = registry.getUserNotifier();
        un.notifyError("Data Loading Failure",
            "Couldn't retrieve SomeData.", exc);
    }
};
```

Note that the observer's code in the example above works just like any other Swing listener. In fact, all events are delivered sequentially and within the Swing event dispatching thread. This means the observer can run synchronously with respect to the UI and will not need to worry about concurrency issues – as long as it runs within Swing. Finally, also note that subsequent feedback events imply computation progress and the DSCallOutcomeEvent is always the last event to be delivered in order of time.

The xxxLoader classes in agents.treeviewer are a good place to look at and see how to use data services view.

### Execution

The next diagram analyzes a concrete call to a view to exemplify the pattern followed by all asynchronous calls in the various views. The call is mapped onto a command, the command is transferred to a processor for asynchronous execution, a handle to the call is returned to the invoker.
Initialization

The `DataViewsFactory` (in `env.data.views`) needs to be initialized before any concrete `BatchCallTree` (in `env.data.views`) is created. The reason for this is that `BatchCallTree`'s constructor needs to cache a reference to the registry so that concrete subclasses can access it later. The `DataViewsFactory` takes care of this initialization task during the container’s start-up procedure by calling `DataViewsFactory.initialize(Container)`. Any data service view should be created in `env.data.views` and declared in `DataViewsFactory.makeNew(Class)`. The method returns an implementation of the corresponding view.

See also:

`Directory contents`

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

15.10 Organization

The source code is organized as follows. All classes share a common base namespace:

`org.openmicroscopy.shoolal`
Two main packages sit under the shoola directory:

- **agents**: All the classes related to concrete agents.
- **env**: All the classes that make up the runtime environment, that is the container.

The **agents** package is further broken down into:

- events
- dataBrowser
- imviewer
- measurement
- metadata
- treeviewer
- util

These packages contain the code for the Data Browser, Data Manager, Viewer, and Measurement agents. The events package contains the events that are used by all these agents.

The **env** package is also broken down into further sub-packages:

- **config**: Registry-related classes.
- **data**: Defines the client’s side interface and implementation of the Remote Facade that we use to access the OMERO server.
- **event**: The event bus classes.
- **init**: Classes to perform initialization tasks at start-up.
- **log**: Adapter classes to wrap log4j\(^{15}\).
- **cache**: Adapter classes to wrap ehcache\(^{16}\).
- **rnd**: The image data provider.
- **ui**: The top frame window and the user notification widgets.

**Note**: Two extra packages are part of the project for convenience reason only:

- **svc**: Provides general services e.g. transport service using HTTP.
- **util**: Collection of classes that be used outside the OMERO structure

**Note**: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

## 15.11 Taskbar

The container provides a top-level window, the taskbar, to let users control some of the container’s tasks like the connection to the remote service or quitting the application. The taskbar contains a menu bar and several toolbars. Agents can add their own entries to the menu bar and to the toolbars in order to trigger agent-specific actions, or can ignore the taskbar altogether. For example, agents that are UIs typically add an entry to the Window menu and to the Quick-launch toolbar (this should be done during the agent’s linking phase) for top-level windows that the user can bring up. Some utility classes provide agents with functionality to link their top-level windows to the taskbar and to manage the display of those windows on screen.

### 15.11.1 Structure and dynamics

The following diagram shows the classes that provide the taskbar service and their relationships:

\(^{15}\)http://logging.apache.org/log4j/

\(^{16}\)http://ehcache.org/
The following diagram shows how some utility classes that can be used to link windows to the TaskBar and to manage their display on screen:

The following diagram shows how the display state of a top window is managed by the TopWindowManager:

### 15.11.2 How to

Agents can use the taskbar directly to add entries to the various menus and toolbars. After retrieving the TaskBar from the Registry, the addXXX and removeXXX are available to do the job.

Agents can add entries to any of the menus within the menu bar – File, Connect, Tasks, Window, Help. Two toolbars, Tasks and Quick-Launch, are also provided for agents to plug in their buttons. Buttons in the Tasks toolbar are usually shortcuts to entries in the Tasks menu. Similarly, buttons in the Quick-Launch toolbar are usually shortcuts to entries in the Window menu.

However, some utility classes provide agents with built-in functionality to link their top-level windows to the taskbar and to manage the display of those windows on screen.

The TopWindow class extends JFrame to provide the base functionality for windows that are to be linked to the TaskBar by means of one quick-launch button and a menu entry in the Window menu. The constructor of this class automatically adds a button to the Quick-Launch toolbar and an entry in the Window menu – subclasses use the configureXXX methods to specify icons, names, and tool tips. These are display-trigger buttons that cause the window to be shown on screen. This class uses the TopWindowManager to control mouse clicks on these buttons as well as to manage the display state of the window.
Figure 15.12: Taskbar utility classes

- how this display state is managed is specified by the TopWindowManager state machine, which is represented in one of the previous diagrams.

Here is an example of a window that inherits from TopWindow:

class MainWindow
    extends TopWindow
{
    //Member fields omitted.

    //Specifies names, icons, and tool tips for the quick-launch button and the
    //window menu entry in the taskbar.
    private void configureDisplayButtons()
    {
        configureQuickLaunchBtn(icons.getIcon("agent.png"),
                               "Display the main window.");
        configureWinMenuEntry("Example Agent", icons.getIcon("agent.png"));
    }
}
OMERO Documentation, Release 5.2.6

Figure 15.13: Taskbar window management

//Builds and lays out this window.
private void buildGUI() { /* Omitted. */ }

//Creates a new instance.
MainWindow(Registry config)
{
    //We have to specify the title of the window to the superclass constructor and pass a reference to the TaskBar, which we get from the Registry.
    super("Example Agent", config.getTaskBar());
    configureDisplayButtons();
    buildGUI();
}
The `TopWindowGroup` class links a group of windows to the `TaskBar` and manages their display on screen. Rather than adding a quick-launch button in the `Quick-Launch` toolbar and an entry in the window menu for each window in the group, the constructor of this class adds a drop-down button (a button that triggers the display of a popup menu) to the `Quick-Launch` toolbar and a sub-menu to the `Window` menu. These menus contain an entry for each window in the group and are populated/depopulated via the add/remove methods. All those menu entries are display-trigger buttons that cause the corresponding window to be shown on screen. This class uses the `TopWindowManager` to control mouse clicks on these buttons as well as to manage the display state of each window in the group.

The following `UIManager` class provides an example of how to use the `TopWindowGroup` class. This example class creates and controls an instance of `MainWindow` (which we have already seen in the previous example) as well as `AuxiliaryWindow` instances. This latter class is just a window which contains two buttons and its code is omitted. `UIManager` delegates to the `TopWindowGroup` class the linkage of `AuxiliaryWindow`'s to the `TaskBar` as well as the management of their display state.

Follows the code:

```java
class UIManager {
    //Inherits from TopWindow, so it is automatically linked to the TaskBar.
    //Contains a button that we listen to. When a mouse click occurs we call
    //createAuxWin().
    private MainWindow mainWindow;

    //Manages all the AuxiliaryWindow's that we have created and not destroyed yet.
    private TopWindowGroup auxWinGroup;

    //Counts how many (ofAux AuxiliaryWindow)‘s that we have created so far.
    private int auxWinCount;

    //Cached reference to access the icons.
    private IconFactory icons;

    //Creates a new instance.
    UIManager(Registry config) {
        auxWinCount = 0;
        icons = (IconFactory) config.lookup("/resources/icons/MyFactory");
        mainWindow = new MainWindow(config);

        //The MainWindow contains a button (not shown in the previous example)
        //which we listen to in order to trigger the creation of new
        //AuxiliaryWindow’s.
        mainWindow.openAuxiliary.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent ae) { createAuxWin(); }
        });

        //The text we pass will be displayed by
        //the sub-menu within the Window menu along with the icon, which will also
        //be the icon displayed by the drop-down button in the Quick-Launch
        //toolbar.
        auxWinGroup = new TopWindowGroup("Aux Win",
            icons.getIcon("edu_languages.png"),
            config.getTaskBar());
    }

    //Creates an AuxiliaryWindow and adds it to the auxWinGroup.
    //Every AuxiliaryWindow contains two buttons, one labeled "Close" and the other
```
private void createAuxWin()
{
    String title = "Aux Window "+(++auxWinCount);
    final AuxiliaryWindow aw = new AuxiliaryWindow(title);

    //Attach listeners and specify actions.
    aw.close.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent ae) {aw.setVisible(false);}
    });
    aw.dispose.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            //Remove from group and dispose.
            auxWinGroup.remove(aw, true);
        }
    });

    //Add to the group. An entry will be added both to the Window sub-menu
    //and to the popup menu triggered by the drop-down button in the
    //Quick-Launch toolbar. We set the display text of those entries to be
    //the same as the window’s title, but we don’t specify any icon.
    auxWinGroup.add(aw, title, null);

    //Bring the window up.
    aw.open();
}

//Releases all UI resources currently in use and returns them to the OS.
void disposeUI()
{
    mainWindow.dispose();
    auxWinGroup.removeAll(true); //Empty group and dispose of all windows.
}
OMERO can be extended by modifying these clients or by writing your own in any of the supported languages.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

16.1 Developing OMERO clients

Note:
- If you are only interested in using our OMERO clients, please see the OMERO clients overview section, which will point you to user guides, demo videos, and download sites.
- This page is intended for developers already familiar with client/server programming. If you are not, your best starting point is to read the Hello World¹ chapter of the Ice manual (or more). A deeper understanding of Ice might not be necessary, but certainly understanding the Ice basics will make reading this guide much easier.

For developers, there are many examples listed below, all of which are stored under: examples² and buildable/runnable via scons³:

```
cd omero-src
./build.py build-all
cd omero-src/examples
python ../target/scons/scons.py
```

Other examples (in Python) can be found here.

16.1.1 Introduction

A Blitz client is any application which uses the OMERO Application Programming Interface to talk to the OMERO.blitz server in any of the supported languages, like Python, C++, Java, or Matlab. A general understanding of the OMERO.server overview may make what is happening behind the scenes more transparent, but is not necessary. The points below outline all that an application writer is expected to know with links to further information where necessary.

16.1.2 Distributed computing

The first hurdle when beginning to work with OMERO is to realize that building distributed-object systems is different from both building standalone clients and writing web applications in frameworks like mod_perl, django, or Ruby on Rails. The remoting framework used by OMERO is Ice⁴ from ZeroC. Ice is comparable to CORBA in many ways, but is typically easier to use.

A good first step is to be aware of the difference between remote and local invocations. Any invocation on a proxy (<class_name>Prx, described below) will result in a call over the network with all the costs that entails. The often-cited

¹https://doc.zeroc.com/display/Ice/Hello+World+Application
²https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/examples
³http://www.scons.org
⁴https://zeroc.com
fallacies of distributed computing\(^5\) all apply, and the developer must be aware of concurrency and latency issues, as well as complete loss of connectivity, all of which we will discuss below.

### 16.1.3 Objects

Before we can begin talking about what you can do with OMERO (the remote method calls available in the OMERO Application Programming Interface), it is helpful to first know what the objects are that we will be distributing. These are the only types that can pass through the API.

**“Slice” mapping language**

Ice provides an interface definition language (IDL)\(^6\) for defining class hierarchies for passing data in a binary format. Similar to WSDL in web services or CORBA’s IDL, slice provides a way to specify how types can pass between different programming languages. For just that reason, several constructs not available in all the supported languages are omitted:

- multiple inheritance (C++ and Python)
- nullable primitive wrappers (e.g. Java’s java.lang.Integer)
- interfaces (Java)
- HashSet types
- iterator types

**Primitives**

Slice defines the usual primitives – `long`, `string`, `bool`, as well as `int`, `double`, and `float` – which map into each language as would be expected. Aliases like “Ice::Long” are available for C++ to handle both 32 and 64 bit architectures.

A simple struct can then be built out of any combination of these types. From `components/blitz/resources/omero/System.ice`\(^7\):

```冰
// The EventContext is all the information the server knows about a
// given method call, including user, read/write status, etc.
class EventContext {
  long userId;
  string userName;
  ...
  bool isAdmin;
  ...
```

**Sequences, dictionaries, enums, and constants**

Other than the “user-defined classes” which we will get to below, slice provides only four built-in building blocks for creating a type hierarchy.

- **Sequences & Dictionaries**: Most of the sequences and dictionaries in use by the OMERO Application Programming Interface are defined in `components/blitz/resources/omero/Collections.ice`\(^8\). Each sequence or dictionary must be defined before it can be used in any class. By default a sequence will map to an array of the given type in Java or a vector in C++, but these mappings can be changed via metadata. (In most cases, a `List` is used in the Java mapping).

- **Constants**: Most of the enumerations for OMERO Application Programming Interface are defined in `components/blitz/resources/omero/Constants.ice`\(^9\). These are values which can be defined once and then referenced in each of the supported programming languages. The only real surprise when working with enumerations is that in Java each constant is mapped to an interface with a single `public final static` field named “value”.

---

\(^6\)http://en.wikipedia.org/wiki/Interface_description_language
\(^7\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/resources/omero/System.ice
\(^8\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/resources/omero/Collections.ice
\(^9\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/resources/omero/Constants.ice
#include <iostream>
#include <omero/Constants.h>
using namespace omero::constants;

int main() {
    std::cout << "By default, no method call can pass more than ";
    std::cout << MESSAGESIZEMAX << "kb" << std::endl;
    std::cout << "By default, client.createSession() will wait ";
    std::cout << (CONNECTTIMEOUT / 1000) << " seconds for a connection" << std::endl;
}

Example: examples/OmeroClients/constants.cpp

sz=omero.constants.MESSAGESIZEMAX.value;
to=omero.constants.CONNECTTIMEOUT.value/1000;
disp(sprintf('By default, no method call can pass more than %d kb',sz));
disp(sprintf('By default, client.createSession() will wait %d seconds for a connection', to));

Example: examples/OmeroClients/constants.m

disp(sprintf('By default, no method call can pass more than %s kb' % MESSAGESIZEMAX)
disp(sprintf('By default, client.createSession() will wait %s seconds for a connection' % (CONNECTTIMEOUT/1000)));

Example: examples/OmeroClients/constants.py

import static omero.rtypes.*
public class constants {
    public static void main(String[] args) {
        System.out.println(String.format("By default, no method call can pass more than %s kb", 
                                           omero.constants.MESSAGESIZEMAX.value));
        System.out.println(String.format("By default, client.createSession() will wait %s seconds for a connection", 
                                           omero.constants.CONNECTTIMEOUT.value/1000));
    }
}

Example: examples/OmeroClients/constants.java

- ** Enums. ** Finally, enumerations which are less used through *OMERO Application Programming Interface*, but which can be useful for simplifying working with constants.

#include <iostream>
#include <omero/Constants.h>
using namespace omero::constants::projection;

int main() {
    std::cout << "IProjection takes arguments of the form: ";
    std::cout << MAXIMUMINTENSITY;
    std::cout << std::endl;
}

Example: examples/OmeroClients/enumerations.cpp

10https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/constants.cpp
11https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/constants.m
12https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/constants.py
13https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/constants.java
14https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/enumerations.cpp
v = omero.constants.projection.ProjectionType.MAXIMUMINTENSITY.value();
disp(sprintf('IProjection takes arguments of the form: %s', v));

Example: examples/OmeroClients/enumerations.m15

import omero
import omero_consts_ice
print "IProjection takes arguments of the form: %s" % omero.constants.projection.ProjectionType.MAXIMUMINTENSITY

Example: examples/OmeroClients/enumerations.py16

Example: examples/OmeroClients/enumerations.java17

RTypes

In Java, the Ice primitives map to non-nullable primitives. And in fact, for the still nullable types java.lang.String as well as all collections and arrays, Ice goes so far as to send an empty string ("") or collection([]) rather than null.

However, the database and OMERO support nullable values and so OMERO.blitz defines a hierarchy of types which wraps the primitives: RTypes18 Since Ice allows references to be nulled, as opposed to primitives, it is possible to send null strings, integers, etc.

#include <omero/RTypesI.h>
using namespace omero::rtypes;
int main() {  
  omero::RStringPtr s = rstring("value");
  omero::RBoolPtr b = rbool(true);
  omero::RLongPtr l = rlong(1);
  omero::RIntPtr i = rint(1);
}

Example: examples/OmeroClients/primitives.cpp19

import omero.rtypes;
a = rtypes.rstring('value');
b = rtypes.rbool(true);
l = rtypes.rlong(1);
i = rtypes.rint(1);

Example: examples/OmeroClients/primitives.m20

---

15 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/enumerations.m
16 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/enumerations.py
17 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/enumerations.java
18 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/resources/omero/RTypes.ice
19 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/primitives.cpp
20 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/primitives.m
from omero.rtypes import *
s = rstring("value")
b = rbool(True)
l = rlong(1)
i = rint(1)

Example: examples/OmeroClients/primitives.py

import static omero.rtypes.*;
public class primitives {
    public static void main(String[] args) {
        omero.RString a = rstring("value");
        omero.RBool b = rbool(true);
        omero.RLong l = rlong(1l);
        omero.RInt i = rint(1);
    }
}

Example: examples/OmeroClients/primitives.java

The same works for collections. The RCollection subclass of RType holds a sequence of any other RType.

#include <omero/RTypesI.h>
using namespace omero::rtypes;
int main() {
    // Sets and Lists may be interpreted differently on the server
    omero::RListPtr l = rlist(); // rstring("a"), rstring("b");
    omero::RSetPtr s = rset(); // rint(1), rint(2));
    // No-varargs (#1242)
}

Example: examples/OmeroClients/rcollection.cpp

% Sets and Lists may be interpreted differently on the server
ja = javaArray('omero.RString',2);
ja(1) = omero.rtypes.rstring('a');
ja(2) = omero.rtypes.rstring('b');
list = omero.rtypes.rlist(ja)
ja = javaArray('omero.RInt',2);
ja(1) = omero.rtypes.rint(1);
ja(2) = omero.rtypes.rint(2);
set = omero.rtypes.rset(ja)

Example: examples/OmeroClients/rcollection.m

import omero
from omero.rtypes import *
% Sets and Lists may be interpreted differently on the server
list = rlist(rstring("a"), rstring("b"));
set = rset(rint(1), rint(2));

Example: examples/OmeroClients/rcollection.py

---

21https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/primitives.py
22https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/primitives.java
23https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/rcollection.cpp
24https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/rcollection.m
25https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/rcollection.py

16.1. Developing OMERO clients
import static omero.rtypes.*;
public class rcollection {
    public static void main(String[] args) {
        // Sets and Lists may be interpreted differently on the server
        omero.RList list = rlist(rstring("a"), rstring("b"));
        omero.RSet set = rset(rint(1), rint(2));
    }
}

Example: examples/OmeroClients/rcollection.java

A further benefit of the RTypes is that they support polymorphism. The original OMERO Application Programming Interface was designed strictly for Java, in which the java.lang.Object type or collections of java.lang.Object could be passed. This is not possible with Ice, since there is no Any type as there is in CORBA.

Instead, omero.RType is the abstract superclass of our “remote type” hierarchy, and any method which takes an “RType” can take any subclass of “RType”.

To allow other types discussed later to also take part in the polymorphism, it is necessary to include RType wrappers for them. This is the category that omero::RObject and omero::RMap fall into.

omero::RTime and omero::RClass fall into a different category. They are identical to omero::RLong and omero::RString, respectively, but are provided as type safe variants.

OMERO model objects

With these components – rtypes, primitives, constants, etc. – it is possible to define the core nouns of OME, the OME-Remote Objects. The OMERO OME-Remote Objects is a translation of the OME XML specification into objects for use by the server, built out of RTypes, sequences and dictionaries, and Details.

Details

The omero.model.Details object contains security and other internal information which does not contain any domain value. Attempting to set any values which are not permitted, will result in a SecurityViolation, for example trying to change the details.owner to the current user.

#include <omero/model/ImageI.h>
#include <omero/model/PermissionsI.h>
using namespace omero::model;
int main() {
    ImagePtr image = new ImageI();
    DetailsPtr details = image->getDetails();
    PermissionsPtr p = new PermissionsI();
    p->setUserRead(true);
    assert(p->isUserRead());
    details->setPermissions(p);
    // Available when returned from server
    // Possibly modifiable
    details->setOwner();
    details->setGroup(new ExperimenterGroupI(1L, false));
    // Available when returned from server
    // Not modifiable
    details->getCreationEvent();
    details->getUpdateEvent();
}

Example: examples/OmeroClients/details.cpp
image = omero.model.ImageI();
details_ = image.getDetails();
p = omero.model.PermissionsI();
p.setUserRead(true);
assert( p.isUserRead() );
details_.setPermissions( p );
% Available when returned from server
% Possibly modifiable
details_.getOwner();
details_.setGroup( omero.model.ExperimenterGroupI(1, false) );
% Available when returned from server
% Not modifiable
details_.getCreationEvent();
details_.getUpdateEvent();

Example: examples/OmeroClients/details.m

import omero
import omero.clients
image = omero.model.ImageI()
details = image.getDetails()
p = omero.model.PermissionsI()
p.setUserRead(True)
assert p.isUserRead()
details.setPermissions(p)
# Available when returned from server
# Possibly modifiable
details.getOwner()
details.setGroup( omero.model.ExperimenterGroupI(1L, False) )
# Available when returned from server
# Not modifiable
details.getCreationEvent()
details.getUpdateEvent()

Example: examples/OmeroClients/details.py

import omero.model.Image;
import omero.model.ImageI;
import omero.model.Details;
import omero.model.Permissions;
import omero.model.PermissionsI;
import omero.model.ExperimenterGroupI;
public class details {
    public static void main(String args[]) {
        Image image = new ImageI();
        Details details = image.getDetails();
        Permissions p = new PermissionsI();
p.setUserRead(true);
        assert p.isUserRead();
        details.setPermissions(p);
        // Available when returned from server
        // Possibly modifiable
details.getOwner();
details.setGroup(new ExperimenterGroupI(1L, false));
        // Available when returned from server
        // Not modifiable
details.getCreationEvent();
details.getUpdateEvent();
    }
}
ObjectFactory and casting

In the previous examples, you may have noticed how there are two classes for each type: `Image` and `ImageI`. Classes defined in slice are by default data objects, more like C++'s `struct`s than anything else. As soon as a class defines a method, however, it becomes an abstract entity and requires application writers to provide a concrete implementation (hence the “I”). All OMERO classes define methods, but OMERO takes care of providing the implementations for you via code generation. For each slice-defined and Ice-generated class `omero.model.Something`, there is an OMERO-generated class `omero.model.SomethingI` which can be instantiated.

```cpp
#include <omero/model/ImageI.h>
#include <omero/model/DatasetI.h>
using namespace omero::model;

int main() {
  ImagePtr image = new ImageI();
  DatasetPtr dataset = new DatasetI(1L, false);
  image->linkDataset(dataset);
}
```

Example: examples/OmeroClients/constructors.cpp

```java
import omero.model.*;
image = ImageI();
dataset = DatasetI(1, false);
image.linkDataset(dataset)
```

Example: examples/OmeroClients/constructors.m

```python
import omero.model.*;
import omero.model.Image;
import omero.model.ImageI;
import omero.model.Dataset;
import omero.model.DatasetI;
import omero.model.DatasetImageLink;
import omero.model.DatasetImageLinkI;
public class constructors {
  public static void main(String args[]) {
    Image image = new ImageI();
    Dataset dataset = new DatasetI(1L, false);
    image.linkDataset(dataset);
  }
}
```

Example: examples/OmeroClients/constructors.py

---

31 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/details.java
32 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/constructors.cpp
33 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/constructors.m
34 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/constructors.py
Example: examples/OmeroClients/constructors.java\textsuperscript{35}

When OME-Remote Objects instances are serialized over the wire and arrive in the client, the Ice runtime must determine which constructor to call. It consults with the ObjectFactory, also provided by OMERO, to create the new classes. If you would like to have your own classes or subclasses created on deserialization, see the Advanced topics section below.

Such concrete implementations provide features which are not available in the solely Ice-based versions. When you would like to use these features, it is necessary to down-cast to the OMERO-based type.

For example, objects in each language binding provide a “more natural” form of iteration for that language.

```cpp
#include <omero/model/ImageI.h>
#include <omero/model/DatasetI.h>
#include <omero/model/DatasetImageLinkI.h>
using namespace omero::model;

int main() {
    ImageIPtr image = new ImageI();
    DatasetIPtr dataset = new DatasetI();
    DatasetImageLinkPtr link = dataset->linkImage(image);
    omero::model::ImageDatasetLinksSeq seq = image->copyDatasetLinks();
    ImageDatasetLinksSeq::iterator beg = seq.begin();
    while(beg != seq.end()) {
        beg++;
    }
}
```

Example: examples/OmeroClients/iterators.cpp\textsuperscript{36}

```cpp
import omero.model.*;
image = ImageI();
dataset = DatasetI();
link = dataset.linkImage(image);
it = image.iterateDatasetLinks();
while it.hasNext() {
    it.next().getChild().getName();
}
```

Example: examples/OmeroClients/iterators.m\textsuperscript{37}

```cpp
import omero
from omero.model.ImageI import ImageI
from omero.model.DatasetI import DatasetI
from omero.model.DatasetImageLinkI import DatasetImageLinkI
image = ImageI()
dataset = DatasetI()
link = dataset.linkImage(image)
for link in image.iterateDatasetLinks():
    link.getChild().getName();
```

Example: examples/OmeroClients/iterators.py\textsuperscript{38}

\textsuperscript{35}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/constructors.java
\textsuperscript{36}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/iterators.cpp
\textsuperscript{37}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/iterators.m
\textsuperscript{38}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/iterators.py
import omero.model.ImageI;
import omero.model.Dataset;
import omero.model.DatasetI;
import omero.model.DatasetImageLink;
import omero.model.DatasetImageLinkI;
import java.util.*;
public class iterators {
    public static void main(String[] args) {
        ImageI image = new ImageI();
        Dataset dataset = new DatasetI();
        DatasetImageLink link = dataset.linkImage(image);
        Iterator<DatasetImageLinkI> it = image.iterateDatasetLinks();
        while (it.hasNext()) {
            it.next().getChild().getName();
        }
    }
}

Example: examples/OmeroClients/iterators.java

Also, each concrete implementation provides static constants of various forms.

#include <omero/model/ImageI.h>
#include <iostream>
int main() {
    std::cout << omero::model::ImageI::NAME << std::endl;
    std::cout << omero::model::ImageI::DATASETLINKS << std::endl;
}

Example: examples/OmeroClients/staticfields.cpp

disp(omero.model.ImageI.NAME);
disp(omero.model.ImageI.DATASETLINKS);

Example: examples/OmeroClients/staticfields.m

import omero
from omero.model.ImageI import ImageI as ImageI
print ImageI.NAME
print ImageI.DATASETLINKS

Example: examples/OmeroClients/staticfields.py

import omero.model.ImageI;
public class staticfields {
    public static void main(String[] args) {
        System.out.println(ImageI.NAME);
        System.out.println(ImageI.DATASETLINKS);
    }
}

Example: examples/OmeroClients/staticfields.java

https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/iterators.java
https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/staticfields.cpp
https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/staticfields.m
https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/staticfields.py
https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/staticfields.java

16.1. Developing OMERO clients
Visibility and loadedness

In the constructor example above, a constructor with two arguments was used to create the Dataset instance linked to the new Image. The Dataset instance so created is considered “unloaded”.

Objects and collections can be created unloaded as a pointer to an actual instance or they may be returned unloaded from the server when they are not actively accessed in a query. Because of the interconnectedness of the OME-Remote Objects, loading one object could conceivably require downloading a large part of the database if there were not some way to “snip-off” sections.

```cpp
#include <omero/model/ImageI.h>
#include <omero/model/DatasetI.h>
#include <omero/ClientErrors.h>
using namespace omero::model;

int main() {
    ImagePtr image = new ImageI(); // A loaded object by default
    assert(image->isLoaded());
    image->unload(); // can then be unloaded
    assert(! image->isLoaded());
    image = new ImageI( 1L, false ); // Creates an unloaded "proxy"
    assert(! image->isLoaded());
    image->getId(); // Ok
    try {
        image->getName(); // No data access is allowed other than id.
    } catch (const omero::ClientError& ce) {
        // Ok.
    }
}
```

Example: examples/OmeroClients/unloaded.cpp

```cpp
import omero
import omero.model

image = omero.model.ImageI(); % A loaded object by default
assert(image.isLoaded());
image.unload(); % can then be unloaded
assert(~ image.isLoaded() );
image = omero.model.ImageI( 1L, false );
assert(~ image.isLoaded() ); % Creates an unloaded "proxy"
image.getId(); % Ok.
try
    image.getName(); % No data access is allowed other than id
catch ME
    % OK
end
```

Example: examples/OmeroClients/unloaded.m

---

44https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/unloaded.cpp
45https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/unloaded.m
Example: examples/OmeroClients/unloaded.py

```python
import omero.model.ImageI;
public class unloaded {
    public static void main(String args[]) {
        ImageI image = new ImageI(); // A loaded object by default
        assert image.isLoaded();
        image.unload(); // can then be unloaded
        assert ! image.isLoaded();
        image = new ImageI(1L, false); // Creates an unloaded "proxy"
        assert ! image.isLoaded();
        image.getId(); // Ok.
        try {  
            image.getName(); // No data access is allowed other than id.
        } catch (Exception e) {
            // Ok.
        }
    }
}
```

Example: examples/OmeroClients/unloaded.java

When saving objects that have unloaded instances in their graph, the server will automatically fill in the values. So, if your `Dataset` contains a collection of `Images`, all of which are unloaded, then they will be reloaded before saving, based on the id. If, however, you had tried to set a value on one of the `Images`, you will get an exception.

To prevent errors when working with unloaded objects, all the `OME-Remote Objects` classes are marked as protected in the slice definitions which causes the implementations in each language to try to hide the fields. In Java and C++ this results in fields with “protected” visibility. In Python, an underscore is prefixed to all the variables. (In the Python case, we have also tried to “strengthen” the hiding of the fields, by overriding `__setattr__`. This is not full proof, but only so much can be done to hide values in Python.)

### Collections

Just as an entire object can be unloaded, any collection field can also be unloaded. However, as mentioned above, since it is not possible to send a null collection over the wire with Ice and working with RTypes can be inefficient, all the `OME-Remote Objects` collections are hidden behind several methods.

```c
#include <omero/model/DatasetI.h>
#include <omero/model/DatasetImageLinkI.h>
#include <omero/model/EventI.h>
#include <omero/model/ImageI.h>
#include <omero/model/PixelsI.h>
using namespace omero::model;
int main(int argc, char* argv[]) {  
   ImagePtr image = new ImageI(1, true);
   image->getDetails()->setUpdateEvent( new EventI(1L, false) );
   // On creation, all collections are initialized to empty, and can be added to.
   assert(image->sizeOfDatasetLinks() == 0);
   DatasetPtr dataset = new DatasetI(1L, false);
   DatasetImageLinkPtr link = image->linkDataset(dataset);
   assert(image->sizeOfDatasetLinks() == 1);
   // If you want to work with this collection, you’ll need to get a copy.
   ImageDatasetLinksSeq links = image->copyDatasetLinks();
   // When you are done working with it, you can unload the datasets, assuming the changes
```
// have been persisted to the server.
image->unloadDatasetLinks();
assert(image->sizeOfDatasetLinks() < 0);
try {
    image->linkDataset( new DatasetI() );
} catch (...) {
    // Can't access an unloaded collection
}
// The reload...() method allows one instance
// to take over a collection from another, if it
// has been properly initialized on the server.
// sameImage will have its collection unloaded.
ImagePtr sameImage = new ImageI(1L, true);
sameImage->getDetails()->setUpdateEvent( new EventI(1L, false) );
sameImage->linkDataset( new DatasetI(1L, false) );
image->reloadDatasetLinks( sameImage );
assert(image->sizeOfDatasetLinks() == 1);
assert(sameImage->sizeOfDatasetLinks() < 0);
// If you would like to remove all the member
// elements from a collection, don’t unload it
// but "clear" it.
image->clearDatasetLinks();
// Saving this to the database will remove
// all dataset links!
// Finally, all collections can be unloaded
// to use an instance as a single row in the database.
image->unloadCollections();
// Ordered collections have slightly different methods.
image = new ImageI(1L, true);
image->addPixels( new PixelsI() );
image->getPixels(0);
image->getPrimaryPixels(); // Same thing
image->removePixels( image->getPixels(0) );
}

Example: examples/OmeroClients/collectionmethods.cpp

import omero.model.*;
image = ImageI(1, true);
image.getDetails().setUpdateEvent( EventI(1, false) );
% On creation, all collections are
% initialized to empty, and can be added
% to.
assert(image.sizeOfDatasetLinks() == 0);
dataset = DatasetI(1, false);
link = image.linkDataset(dataset);
assert(image.sizeOfDatasetLinks() == 1);
% If you want to work with this collection,
% you’ll need to get a copy.
links = image.copyDatasetLinks();
% When you are done working with it, you can
% unload the datasets, assuming the changes
% have been persisted to the server.
image.unloadDatasetLinks();
assert(image.sizeOfDatasetLinks() < 0);
try
    image.linkDataset( DatasetI() );
catch ME
    % Can’t access an unloaded collection
end
% The reload...() method allows one instance

---

https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/collectionmethods.cpp
% to take over a collection from another, if it
% has been properly initialized on the server.
% sameImage will have its collection unloaded.
sameImage = ImageI(l, true);
sameImage.getDetails().setUpdateEvent( EventI(l, false) );
sameImage.linkDataset( DatasetI(l, false) );
image.reloadDatasetLinks( sameImage );
assert(image.sizeOfDatasetLinks() == l);
assert(sameImage.sizeOfDatasetLinks() < 0);
% If you would like to remove all the member
% elements from a collection, don’t unload it
% but ”clear” it.
image.clearDatasetLinks();
% Saving this to the database will remove
% all dataset links!
% Finally, all collections can be unloaded
% to use an instance as a single row in the database.
image.unloadCollections();
% Ordered collections have slightly different methods.
image = ImageI(l, true);
image.addPixels( PixelsI() );
image.getPixels(0);
image.getPrimaryPixels(); % Same thing
image.removePixels( image.getPixels(0) );

Example: examples/OmeroClients/collectionmethods.m

```matlab
import omero
import omero.clients
ImageI = omero.model.ImageI
DatasetI = omero.model.DatasetI
EventI = omero.model.EventI
PixelsI = omero.model.PixelsI
image = ImageI(long(l), True)
image.getDetails().setUpdateEvent( EventI(1L, False) )
# On creation, all collections are
# initialized to empty, and can be added
# to.
assert image.sizeOfDatasetLinks() == 0
dataset = DatasetI(long(l), False)
link = image.linkDataset(dataset)
assert image.sizeOfDatasetLinks() == 1
# If you want to work with this collection,
# you’ll need to get a copy.
links = image.copyDatasetLinks()
# When you are done working with it, you can
# unload the datasets, assuming the changes
# have been persisted to the server.
image.unloadDatasetLinks()
assert image.sizeOfDatasetLinks() < 0
try:
    image.linkDataset( DatasetI() )
except:
    # Can’t access an unloaded collection
    pass
# The reload...() method allows one instance
# to take over a collection from another, if it
# has been properly initialized on the server.
# sameImage will have its collection unloaded.
sameImage = ImageI(lL, True)
sameImage.getDetails().setUpdateEvent( EventI(ll, False) )
```

49https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/collectionmethods.m
sameImage.linkDataset( DatasetI(long(1), False) )
image.reloadDatasetLinks( sameImage )

assert image.sizeOfDatasetLinks() == 1
assert sameImage.sizeOfDatasetLinks() < 0
# If you would like to remove all the members
# elements from a collection, don’t unload it
# but “clear” it.
image.clearDatasetLinks()
# Saving this to the database will remove
# all dataset links!
# Finally, all collections can be unloaded
# to use an instance as a single row in the database.
image.unloadCollections()
# Ordered collections have slightly different methods.
image = ImageI(long(1), True)
image.addPixels( PixelsI() )
image.getPixels(0)
image.getPrimaryPixels() # Same thing
image.removePixels( image.getPixels(0) )

Example: examples/OmeroClients/collectionmethods.py

import omero.model.Dataset;
import omero.model.DatasetI;
import omero.model.DatasetImageLink;
import omero.model.DatasetImageLinkI;
import omero.model.EventI;
import omero.model.Image;
import omero.model.ImageI;
import omero.model.Pixels;
import omero.model.PixelsI;
import java.util.*;
public class collectionmethods {
    public static void main(String args[]) {
        Image image = new ImageI(1, true);
        image.getDetails().setUpdateEvent( new EventI(1L, false) );
        // On creation, all collections are
        // initialized to empty, and can be added
        // to.
        assert image.sizeOfDatasetLinks() == 0;
        Dataset dataset = new DatasetI(1L, false);
        DatasetImageLink link = image.linkDataset(dataset);
        assert image.sizeOfDatasetLinks() == 1;
        // If you want to work with this collection,
        // you’ll need to get a copy.
        List<DatasetImageLink> links = image.copyDatasetLinks();
        // When you are done working with it, you can
        // unload the datasets, assuming the changes
        // have been persisted to the server.
        image.unloadDatasetLinks();
        assert image.sizeOfDatasetLinks() < 0;
        try {
            image.linkDataset( new DatasetI() );
        } catch (Exception e) {
            // Can’t access an unloaded collection
            // The reload...() method allows one instance
            // to take over a collection from another, if it
            // has been properly initialized on the server.
            // sameImage will have its collection unloaded.
            Image sameImage = new ImageI(1L, true);
        }
    }
}
sameImage.getDetails().setUpdateEvent( new EventI(1L, false) );
sameImage.linkDataset( new DatasetI(1L, false) );
image.reloadDatasetLinks( sameImage );
assert image.sizeOfDatasetLinks() == 1;
assert sameImage.sizeOfDatasetLinks() < 0;
// If you would like to remove all the member
// elements from a collection, don’t unload it
// but “clear” it.
image.clearDatasetLinks();
// Saving this to the database will remove
// all dataset links!
// Finally, all collections can be unloaded
// to use an instance as a single row in the database.
image.unloadCollections();
// Ordered collections have slightly different methods.
image = new ImageI(1L, true);
image.addPixels( new PixelsI() );
image.getPixels(0);
image.getPrimaryPixels(); // Same thing
image.removePixels( image.getPixels(0) );
}
}

Example: examples/OmeroClients/collectionmethods.java

These methods prevent clients from accessing the collections directly, and any improper access will lead to an
izer.Error.

Interfaces

As mentioned above, one of the Java features which is missing from the slice definition language is the ability to have concrete
classes implement multiple interfaces. Much of the OME-Remote Objects in the RMI-based types (ome.model) was based on
the use of interfaces.

* IObject is the root interface for all object types. Methods: getId(), getDetails(),...

* IEnum is an enumeration value. Methods: getValue()

* ILink is a link between two other types. Methods: getParent(), getChild()

* IMutable is an instance for changes will be persisted. Methods: getVersion()

Instead, the Ice-based types (omeromodel) all subclass from the same concrete type – omeromodel.IObject – and it
has several methods defined for testing which of the omeromodel interfaces are implemented by any type.

Use of such methods is naturally less object-oriented and requires if/then blocks, but within the confines of the mapping language
is a next-best option.

# No cpp example

import omeromodel.*;
o = EventI();
assert( ~ o.isMutable() );
o = ExperimenterI();
assert( o.isMutable() );
assert( o.isGlobal() );

---

51https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/collectionmethods.java
52https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/model/src/ome/model/IObject.java
53https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/model/src/ome/model/IEnum.java
54https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/model/src/ome/model/ILink.java
55https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/model/src/ome/model/IMutable.java
assert( o.isAnnotated() );
o = GroupExperimenterMapI();
assert( o.isLink() );
someObject = ExperimenterI();
% Some method call and you no longer know what someObject is
if (~ someObject.isMutable() )
    % No need to update
elseif (someObject.isAnnotated())
    % deleteAnnotations(someObject);
end

Example: examples/OmeroClients/interfaces.m

import omero
from omero_model_EventI import EventI
from omero_model_ExperimenterI import ExperimenterI
from omero_model_GroupExperimenterMapI import GroupExperimenterMapI
assert ( not EventI().isMutable() )
assert ExperimenterI().isMutable()
assert ExperimenterI().isGlobal()
assert ExperimenterI().isAnnotated()
assert GroupExperimenterMapI().isLink()

Example: examples/OmeroClients/interfaces.py

import omero.model.IObject;
import omero.model.EventI;
import omero.model.ExperimenterI;
import omero.model.GroupExperimenterMapI;
public class interfaces {
    public static void main(String args[]) {
        assert ! new EventI().isMutable();
        assert new ExperimenterI().isMutable();
        assert new ExperimenterI().isGlobal();
        assert new ExperimenterI().isAnnotated();
        assert new GroupExperimenterMapI().isLink();
        IObject someObject = new ExperimenterI();
        // Some method call and you no longer know what someObject is
        if ( ! someObject.isMutable() ) {
            // No need to update
        } else if (someObject.isAnnotated()) {
            // deleteAnnotations(someObject);
        }
    }
}

Example: examples/OmeroClients/interfaces.java

Improvement of this situation by adding abstract classes is planned. However, the entire functionality will not be achievable because of single inheritance.

Language-specific behavior

Smart pointers (C++ only)

An important consideration when working with C++ is that the OME-Remote Objects classes themselves have no copy-constructors and no assignment operator (operator=), and so cannot be allocated on the stack. Combined with smart pointers this effectively

---

56 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/interfaces.m
57 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/interfaces.py
58 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/interfaces.java
prevents memory leaks.

The code generated types must be allocated on the heap with `new` and used in combination with the smart pointer typedefs which handle calling the destructor when the reference count hits zero.

```cpp
#include <omero/model/ImageI.h>
using namespace omero::model;

int main()
{
    // ImageI image(); // ERROR
    // ImageI image = new ImageI(); // ERROR
    ImageIPtr image1 = new ImageI(); // OK
    ImageIPtr image2(new ImageI()); // OK
    // image1 pointer takes value of image2
    // image1's content is garbage collected
    image1 = image2;
    //
    // Careful with boolean contexts
    //
    if (image1 && image1 == 1) {
        // Means non-null
        // This object can be dereferenced
    }
    ImageIPtr nullImage; // No assignment
    if ( !nullImage && nullImage == 0) {
        // Dereferencing nullImage here would throw an exception:
        // nullImage->getId(); // IceUtil::NullHandleException!
    }
}
```

Example: `examples/OmeroClients/smartpointers.cpp`

```
# No m example

# No py example

# No java example
```

### Warning

As shown in the example, using a smart pointer instance in a boolean or integer/long context, returns 1 for true (i.e. non-null) or 0 for false (i.e. null). Be especially careful with the RTypes.

For more information, see 6.14.6 Smart Pointers for Classes\(^{60}\) in the Ice manual, which also describes the `Ice.GC.Interval` parameter which determines how often garbage collection runs in C++ to reap objects. This is necessary with the OME-Remote Objects since there are inherently cycles in the object graph.

Another point type which may be of use is `omero::client_ptr`. It also performs reference counting and will call `client.closeSession()` once the reference count hits zero. Without `client_ptr`, your code will need to be surrounded by a try/catch block. Otherwise, 1) sessions will be left open on the server, and 2) your client may hang on exit.

```cpp
#include <omero/client.h>
int main(int argc, char* argv[]) {
    // Duplicating the argument list. ticket:1246
    Ice::StringSeq args1 = Ice::argsToStringSeq(argc, argv);
    Ice::StringSeq args2(args1);
```

\(^{59}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/smartpointers.cpp

\(^{60}\)https://doc.zeroc.com/display/Ice/Smart+Pointers+for+Classes
Ice::InitializationData id1, id2;
id1.properties = Ice::createProperties(args1);
id2.properties = Ice::createProperties(args2);
// Either
omero::client client(id1);
try {
    // Do something like
    // client.createSession();
} catch (…) {
    client.closeSession();
}
//
// Or
//
{
    omero::client_ptr client = new omero::client(id2);
    // Do something like
    // client->createSession();
}
// Client was destroyed via RAII

Example: examples/OmeroClients/clientpointer.cpp

# No m example

# No py example

# No java example

__getattr__ and __setattr__ (Python only)

Like smart pointers for OMERO C++ language bindings, the OMERO Python language bindings SDK defines __getattr__ and __setattr__ methods for all OME-Remote Objects classes. Rather than explicitly calling the getFoo() and setFoo() methods, field-like access can be used. (It should be noted, however, that the accessors will perform marginally faster.)

# No cpp example

# No m example

```python
import omero
import omero.clients
from omero.rtypes import *
i = omero.model.ImageI()
# # Without __getattr__ and __setattr__
# i.setName( rstring("name") )
assert i.getName().getValue() == "name"
# # With __getattr__ and __setattr__
```

61https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/clientpointer.cpp
```python
i = omero.model.ImageI()
i.name = rstring("name")
assert i.name.val == "name"

# Collections, however, cannot be accessed via the special methods due to the dangers outlined above

try:
    i.datasetLinks[0]
except AttributeError, ae:
    pass
```

Example: examples/OmeroClients/getsetattr.py

# No java example

### Method inspection and code completion (Matlab & Python)

Ice generates a number of internal (private) methods which are not intended for general consumption. Unfortunately, Matlab’s code-completion as well as Python’s `dir` method return these methods, which can lead to confusion. In general, the API user can ignore any method beginning with an underscore or with `ice_`. For example,

```python
>>> for i in dir(omero.model.ImageI):
...     if i.startswith("_") or i.startswith("ice_")):
...         print i
...     (snip)
_op_addAllDatasetImageLinkSet
_op_addAllImageAnnotationLinkSet
_op_addAllPixelsSet
_op_addAllRoiSet
_op_addAllWellSampleSet
... ice_id
ice_ids
ice_isA
ice_ping
ice_postUnmarshal
ice_preMarshal
ice_staticId
ice_type
>>>```

## 16.1.4 Services overview

After discussing the many types and how to create them, the next obvious question is what one can actually do with them. For that, we have to look at what services are provided by `OMERO.blitz`, how they are obtained, used, and cleaned up.

### OMERO client configuration

The first step in accessing the `OMERO Application Programming Interface` and therefore the first thing to plan when writing an OMERO client is the proper configuration of an `omero.client` instance. The `omero.client` (or in C++ `omero::client`) class tries to wrap together and simplify as much of working with Ice as possible. Where it can, it imports or `<#includes>` types for you,

---

62https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/getsetattr.py
creates an Ice.Communicator and registers an ObjectFactory. Typically, the only work on the client developers part is to properly configure the omero.client object and then login.

In the simplest case, configuration requires only the server host, username, and password with which you want to login. But as you can see below, there are various ways to configure your client, and this is really only the beginning.

```
#include <omero/client.h>
#include <iostream>

int main(int argc, char* argv[]) {
    // All configuration in file pointed to by
    // --Ice.Config=file.config
    // No username, password entered
    try {
        omero::client client1(argc, argv);
        client1.createSession();
        client1.closeSession();
    } catch (const Glacier2::PermissionDeniedException& pd) {
        // Bad password?
    } catch (const Ice::ConnectionRefusedException& cre) {
        // Bad address or port?
    }

    // Most basic configuration.
    // Uses default port 4064
    // createSession needs username and password
    try {
        omero::client client2("localhost");
        client2.createSession("root", "ome");
        client2.closeSession();
    } catch (const Glacier2::PermissionDeniedException& pd) {
        // Bad password?
    } catch (const Ice::ConnectionRefusedException& cre) {
        // Bad address or port?
    }

    // Configuration with port information
    try {
        omero::client client3("localhost", 24063);
        client3.createSession("root", "ome");
        client3.closeSession();
    } catch (const Glacier2::PermissionDeniedException& pd) {
        // Bad password?
    } catch (const Ice::ConnectionRefusedException& cre) {
        // Bad address or port?
    }

    // Advanced configuration in C++ takes place
    // via an InitializationData instance.
    try {
        Ice::InitializationData data;
        data.properties = Ice::createProperties();
        data.properties->setProperty("omero.host", "localhost");
        omero::client client4(data);
        client4.createSession("root", "ome");
        client4.closeSession();
    } catch (const Glacier2::PermissionDeniedException& pd) {
        // Bad password?
    } catch (const Ice::ConnectionRefusedException& cre) {
        // Bad address or port?
    }

    // std::map to be added (ticket:1278)
    try {
        Ice::InitializationData data;
        data.properties = Ice::createProperties();
        data.properties->setProperty("omero.host", "localhost");
        data.properties->setProperty("omero.user", "root");
        data.properties->setProperty("omero.pass", "ome");
        omero::client client5(data);
    }
```
// Again, no username or password needed
// since present in the data. But they *can*
// be overridden.
client5.createSession();
client5.closeSession();
} catch (const Glacier2::PermissionDeniedException& pd) {
  // Bad password?
} catch (const Ice::ConnectionRefusedException& cre) {
  // Bad address or port?
}

Example: examples/OmeroClients/configuration.cpp

% All configuration in file pointed to by
% --Ice.Config=file.config
% No username, password entered
args = javaArray('java.lang.String',1);
args(1) = java.lang.String('--Ice.Config=ice.config');
client1 = omero.client(args);
client1.createSession();
client1.closeSession();
% Most basic configuration.
% Uses default port 4064
% createSession needs username and password
client2 = omero.client('localhost');
client2.createSession('root', 'ome');
client2.closeSession();
% Configuration with port information
client3 = omero.client('localhost', 10463);
client3.createSession('root', 'ome');
client3.closeSession();
% Advanced configuration can also be done
% via an InitializationData instance.
data = Ice.InitializationData();
data.properties = Ice.Util.createProperties();
data.properties.setProperty('omero.host', 'localhost');
client4 = omero.client(data);
client4.createSession('root', 'ome');
client4.closeSession();
% Or alternatively via a java.util.Map instance
map = java.util.HashMap();
map.put('omero.host', 'localhost');
map.put('omero.user', 'root');
map.put('omero.pass', 'ome');
client5 = omero.client(map);
% Again, no username or password needed
% since present in the map. But they *can*
% be overridden.
client5.createSession();
client5.closeSession();

Example: examples/OmeroClients/configuration.m

import omero
import Ice

# All configuration in file pointed to by
# --Ice.Config=file.config or ICE_CONFIG
# environment variable;

63https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/configuration.cpp
64https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/configuration.m
try:
    client1 = omero.client()
    client1.createSession()
    client1.closeSession()
except Ice.ConnectionRefusedException:
    pass  # Bad address or port?

# Most basic configuration.
# Uses default port 4064.
# createSession needs username and password
try:
    client2 = omero.client("localhost")
    client2.createSession("root","ome")
    client2.closeSession()
except Ice.ConnectionRefusedException:
    pass  # Bad address or port?

# Configuration with port information
try:
    client3 = omero.client("localhost", 24064)
    client3.createSession("root","ome")
    client3.closeSession()
except Ice.ConnectionRefusedException:
    pass  # Bad address or port?

# Advanced configuration can also be done
# via an InitializationData instance.
data = Ice.InitializationData()
data.properties = Ice.createProperties()
data.properties.setProperty("omero.host", "localhost")
try:
    client4 = omero.client(data)
    client4.createSession("root","ome")
    client4.closeSession()
except Ice.ConnectionRefusedException:
    pass  # Bad address or port?

# Or alternatively via a dict instance
m = {"omero.host":"localhost",
    "omero.user":"root",
    "omero.pass":"ome"}
client5 = omero.client(m)
# Again, no username or password needed
# since present in the map. But they *can* # be overridden.

Example: examples/OmeroClients/configuration.py

---

65 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/configuration.py
client1.closeSession();
}
// Most basic configuration.
// Uses default port 4064
// createSession needs username and password
omero.client client2 = new omero.client("localhost");
try {
    client2.createSession("root", "ome");
} catch (Ice.ConnectionRefusedException cre) {
    // Bad address or port?
} finally {
    client2.closeSession();
}
// Configuration with port information
omero.client client3 = new omero.client("localhost", 4064);
try {
    client3.createSession("root", "ome");
} catch (Ice.ConnectionRefusedException cre) {
    // Bad address or port?
} finally {
    client3.closeSession();
}
// Advanced configuration can also be done
// via an InitializationData instance.
Ice.InitializationData data = new Ice.InitializationData();
data.properties = Ice.Util.createProperties();
data.properties.setProperty("omero.host", "localhost");
omero.client client4 = new omero.client(data);
try {
    client4.createSession("root", "ome");
} catch (Ice.ConnectionRefusedException cre) {
    // Bad address or port?
} finally {
    client4.closeSession();
}
// Or alternatively via a java.util.Map instance
java.util.Map<String, String> map = new java.util.HashMap<String, String>();
map.put("omero.host", "localhost");
map.put("omero.user", "root");
map.put("omero.pass", "ome");
omero.client client5 = new omero.client(map);
// Again, no username or password needed
// since present in the map. But they *can*
// be overridden.
try {
    client5.createSession();
} catch (Ice.ConnectionRefusedException cre) {
    // Bad address or port?
} finally {
    client5.closeSession();
}

Example: examples/OmeroClients/configuration.java

To find out more about using the Ice.Config file for configuration, see etc/templates/ice.config.

---

66https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/configuration.java
67https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/etc/templates/ice.config
What is a ServiceFactory?

In each of the examples above, the result of configuration was the ability to call `createSession` which returns a `ServiceFactoryPrx`.

The ServiceFactory is the clients representation of the user’s server-side session, which multiple clients can connect to it simultaneously. A `ServiceFactoryPrx` object is acquired on login via the `createSession` method, and persists until either it is closed or a timeout is encountered unless additional clients attach to it. This is done via `client.joinSession(String uuid)`. In that case, the session is not finally closed until its reference count drops to zero.

It produces services!

Once a client has been configured properly, and has an active in ServiceFactory in hand, it is time to start accessing services. The collection of all services provided by OMERO is known as the OMERO Application Programming Interface. Each service is defined in a slice file under `components/blitz/resources/omero`\(^{68}\), along with the definition of ServiceFactory itself:

```java
interface ServiceFactory extends Glacier2::Session
{
    // Central OMERO.blitz stateless services.
    IAdmin* getAdminService() throws ServerError;
    IConfig* getConfigService() throws ServerError;

    // Central OMERO.blitz stateful services.
    Gateway* createGateway() throws ServerError;
    ...
}
```

In the definition above, the return values look like C/C++ pointers, which in Ice’s definition language represents return-by-proxy. When a client calls, `serviceFactory.getAdminService()` it will receive an `IAdminPrx`. **Any call on that object is a remote invocation.**

Stateless vs. stateful

Most methods on the ServiceFactory return either a stateless or a stateful service factory. Stateless services are those returned by calls to “getSomeNameService()”. They implement `omero.api.ServiceInterface` but not its subinterface `omero.api.StatefulServiceInterface`. Stateless services are for all intents and purposes singletons, though the implementation may vary.

Stateful services are returned by calls to “createSomething()” and implement `omero.api.StatefulServiceInterface`. Each maintains a state machine with varying rules on initialization and usage. It is important to guarantee that calls are ordered as described in the documentation for each stateful service. **It is also important to always close stateful services to free up server resources.** If you fail to manually call `StatefulServiceInterfacePrx.close()`, it will be called for you on session close/timeout.

What are timeouts?

The following code has a resource leak:

```python
import omero, sys
c = omero.client()
s = c.createSession()
sys.exit(0)
```

Although the client will not suffer any consequences, this snippet leaves a session open on the server. If the server failed to eventually reap such sessions, they would eventually consume all available memory. To get around this, the server implements

\(^{68}\)https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/blitz/resources/omero

\(^{69}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/resources/omero/API.ice
timeouts on all sessions. **It is the clients responsibility to periodically contact the server to keep the session alive.** Since threading policies vary in applications, no strict guideline is available on how to do this. Almost any API method will suffice to tell the server that the client is still active. Important is that the call happens within every timeout window.

```python
# No cpp example

# No m example

import time
import omero
import threading
IDLETIME = 5
c = omero.client()
s = c.createSession()
re = s.createRenderingEngine()
class KeepAlive(threading.Thread):
    def run(self):
        self.stop = False
        while not self.stop:
            time.sleep(IDLETIME)
            print "calling keep alive"
            try:
                s.keepAllAlive([re])
            except:
                c.closeSession()
raise
keepAlive = KeepAlive()
keepAlive.start()
time.sleep(IDLETIME * 2)
keepAlive.stop = True
```

Example: examples/OmeroClients/timeout.py

```java
import omero.*;
import omero.api.*;
import omero.model.*;
import omero.sys.*;
public class timeout {
    static int IDLETIME = 5;
    static client c;
    static ServiceFactoryPrx s;
    public static void main(String[] args) throws Exception {
        final int idletime = args.length > 1 ? Integer.parseInt(args[0]) : IDLETIME;
        c = new client(args);
        s = c.createSession();
        System.out.println(s.getAdminService().getEventContext().sessionUuid);
        final RenderingEnginePrx re = s.createRenderingEngine(); // for keep alive
        class Run extends Thread {
            public boolean stop = false;
            public void run() {
                while (! stop ) {
                    try {
                        Thread.sleep(idletime*1000L);
                    } catch (Exception e) {
                        // ok
                    }
                }
            }
        }
```

https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/timeout.py
System.out.println(System.currentTimeMillis() + " calling keep alive");
try {
    // Currently, passing a null or empty array to keepAllAlive
    // would suffice. For future-proofing, however, it makes sense
    // to pass stateful services.
    s.keepAllAlive(new ServiceInterfacePrx[]{re});
} catch (Exception e) {
    c.closeSession();
    throw new RuntimeException(e);
}

final Run run = new Run();
class Stop extends Thread {
    public void run() {
        run.stop = true;
    }
}
Runtime.getRuntime().addShutdownHook(new Stop());
run.start();

Example: examples/OmeroClients/timeout.java

Exceptions

Probably the most critical thing to realize is that any call on a proxy, which includes ServiceFactoryPrx or any of the *Prx service classes is a remote invocation on the server. Therefore proper exception handling is critical. The definition of the various exceptions is outlined on the Exception handling page and so will not be repeated here. However, how are these sensibly used?

One easy rule is that every omero.client object which you successfully call createSession() on must have closeSession() called on it before you exit.

```java
omeroclient client = new omeroclient();
client.createSession();
try {
    // do whatever you want
} finally {
    client.closeSession();
}
```

Obviously, the work you do in your client will be much more complicated, and may be under layers of application code. But when designing where active omeroclient objects are kept, be sure that your clean-up code takes care of them.

### 16.1.5 IQuery

Now that we have a good idea of the basics, it might be interesting to start asking the server what it has got. The most powerful way of doing this is by using IQuery and the Hibernate Query Language (HQL).

```c++
#include <omero/api/IQuery.h>
#include <omero/client.h>
#include <omero/RTypesI.h>
#include <omero/sys/ParametersI.h>
using namespace omero::rtypes;
```

71https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/timeout.java
int main(int argc, char* argv[]) {
    omero::client_ptr client = new omero::client(argc, argv);
    omero::api::IServiceFactoryPrx sf = client->createSession();
    omero::api::IQueryPrx q = sf->getQueryService();
    std::string query_string = "select i from Image i where i.id = :id and name like :namedParameter";
    omero::sys::ParametersIPtr p = new omero::sys::ParametersI();
    p->add("id", rlong(1L));
    p->add("namedParameter", rstring("cell%mit%"));
    omero::api::IObjectList results = q->findAllByQuery(query_string, p);
}

Example: examples/OmeroClients/queries.cpp72

[client,sf] = loadOmero;
try
    q = sf.getQueryService();
    query_string = 'select i from Image i where i.id = :id and name like :namedParameter';
    p = omero.sys.ParametersI();
    p.add('id', omero.rtypes.rlong(1));
    p.add('namedParameter', omero.rtypes.rstring('cell%mit%'));
    results = q.findAllByQuery(query_string, p) % java.util.List
catch ME
    client.closeSession();
end

Example: examples/OmeroClients/queries.m73

import sys
import omero
from omero.rtypes import *
from omero.sys_ParametersI import ParametersI
client = omero.client(sys.argv)
try:
    sf = client.createSession()
    q = sf.getService() 
    query_string = "select i from Image i where i.id = :id and name like :namedParameter";
    p = ParametersI()
    p.addId(1L)
    p.add("namedParameter", rstring("cell%mit%"));
    results = q.findAllByQuery(query_string, p)
finally:
    client.closeSession()

Example: examples/OmeroClients/queries.py74

import java.util.List;
import static omero.rtypes.*;
import omero.api.ServiceFactoryPrx;
import omero.api.IQueryPrx;
import omero.model.IObject;
import omero.model.ImageI;
import omero.model.PixelsI;
import omero.sys.ParametersI;
public class queries {
    public static void main(String args[]) throws Exception {
        omero.client client = new omero.client(args);
        try {

72 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/queries.cpp
73 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/queries.m
74 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/queries.py
ServiceFactoryPrx sf = client.createSession();
IQueryPrx q = sf.getQueryService();
String query_string = "select i from Image i where i.id = :id and name like :namedParameter"
ParametersI p = new ParametersI();
p.add("id", rlong(1L));
p.add("namedParameter", rstring("cell%mit%"));
List<IObject> results = q.findAllByQuery(query_string, p);
}
finally {
    client.closeSession();
}

Example: examples/OmeroClients/queries.java

The query_string is an example of HQL. It looks a lot like SQL, but works with objects and fields rather than tables and columns (though in OMERO these are usually named the same). The Parameters object allow for setting named parameters (:id) in the query to allow for re-use, and is the only other argument need to IQueryPrx.findAllByQuery() to get a list of IObject instances back. They are guaranteed to be of type omero::model::Image, but you may have to cast them to make full use of that information.

16.1.6 IUpdate

After you have successfully read objects, an obvious thing to do is create your own. Below is a simple example of creating an image object:

#include <IceUtil/Time.h>
#include <omero/api/IUpdate.h>
#include <omero/client.h>
#include <omero/RTypesI.h>
#include <omero/model/ImageI.h>
using namespace omero::rtypes;
int main(int argc, char* argv[]) {
    omero::client_ptr client = new omero::client(argc, argv);
    omero::model::ImageI i = new omero::model::ImageI();
    i->setName( rstring("name") );
    i->setAcquisitionDate( rtime(IceUtil::Time::now().toMilliSeconds()) );
    omero::api::ServiceFactoryPrx sf = client->createSession();
    omero::api::IUpdatePrx u = sf->getUpdateService();
    i = omero::model::ImagePtr::dynamicCast( u->saveAndReturnObject( i ) );
}

Example: examples/OmeroClients/updates.cpp

[client,sf] = loadOmero;
try
    i = omero.model.ImageI();
    i.setName(omero.rtypes.rstring('name'));
    i.setAcquisitionDate(omero.rtypes.rtime(java.lang.System.currentTimeMillis()));
    u = sf.getUpdateService();
    i = u.saveAndReturnObject( i )
    disp(i.getId().getValue());
catch ME
    disp(ME);
    client.closeSession();
end

75 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/queries.java
76 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/updates.cpp
Example: examples/OmeroClients/updates.m\(^77\)

```m
import sys
import time
import omero
import omero.clients
from omero.rtypes import *
client = omero.client(sys.argv)
try:
i = omero.model.ImageI()
i.name = rstring("name")
i.acquisitionDate = rtime(time.time() * 1000)
sf = client.createSession()
u = sf.getUpdateService()
i = u.saveAndReturnObject( i )
finally:
    client.closeSession()
```

Example: examples/OmeroClients/updates.py\(^78\)

```python
import java.util.List;
import static omero.rtypes.*;
import omero.api.ServiceFactoryPrx;
import omero.api.IUpdatePrx;
import omero.model.ImageI;
import omero.model.Image;
public class updates {
    public static void main(String args[]) throws Exception {
        omero.client client = new omero.client(args);
        try {
            Image i = new ImageI();
i.setName( rstring("name") );
i.setAcquisitionDate( rtime(System.currentTimeMillis()) );
            ServiceFactoryPrx sf = client.createSession();
            IUpdatePrx u = sf.getUpdateService();
i = (Image) u.saveAndReturnObject( i );
        } finally {
            client.closeSession();
        }
    }
}
```

Example: examples/OmeroClients/updates.java\(^79\)

### 16.1.7 Examples

To tie together some of the topics which we have outlined above, we would like to eventually have several more or less complete application examples which you can use to get started. For the moment, there is just one simpler example `TreeList`, but more will certainly be added. Let us know any ideas you may have.

**TreeList**

# No cpp example

---

\(^77\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/updates.m

\(^78\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/updates.py

\(^79\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/updates.java
function projects = AllProjects(query, username)
q = ['select p from Project p join fetch p.datasetLinks dil ',
'join fetch dil.child where p.details.owner.omeName = :name'];
p = omero.sys.ParametersI();
p.add('name', omero.rtypes.rstring(username));
projects = query.findAllByQuery(q, p);

Example: examples/TreeList/AllProjects.m

import omero
from omero.rtypes import *
from omero.sys_ParametersI import ParametersI
def getProjects(query_prx, username):
    return query_prx.findAllByQuery("select p from Project p join fetch p.datasetLinks dil join fetch dil.child where p.details.owner.omeName = :name", ParametersI().add("name", rstring(username)))

Example: examples/TreeList/AllProjects.py

import java.util.List;
import omero.model.Project;
import omero.api.IQueryPrx;
import omero.sys.ParametersI;
import static omero.rtypes.*;
public class AllProjects {
    public static List<Project> getProjects(IQueryPrx query, String username) throws Exception {
        List rv = query.findAllByQuery("select p from Project p join fetch p.datasetLinks dil join fetch dil.child where p.details.owner.omeName = :name", new ParametersI().add("name", rstring(username)));
        return (List<Project>) rv;
    }
}

Example: examples/TreeList/AllProjects.java

# No cpp example

function PrintProjects(projects)
if (projects.size()==0)
    return;
end;
for i=0:projects.size()-1,
    project = projects.get(i);
    disp(project.getName().getValue());
    links = project.copyDatasetLinks();
    if (links.size()==0)
        return;
end;
for j=0:links.size()-1,
    pdl = links.get(j);
    dataset = pdl.getChild();
    disp(sprintf(' %s', char(dataset.getName().getValue())));
end;
end;

80) https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/TreeList/AllProjects.m
81) https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/TreeList/AllProjects.py
82) https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/TreeList/AllProjects.java
Example: examples/TreeList/PrintProjects.m \(^{83}\)

```python
def print_(projects):
    for project in projects:
        print project.getName().val
        for pdl in project.copyDatasetLinks():
            dataset = pdl.getChild()
            print "  " + dataset.getName().val
```

Example: examples/TreeList/PrintProjects.py \(^{84}\)

```python
import java.util.List;
import omero.model.Project;
import omero.model.ProjectDatasetLink;
import omero.model.Dataset;
public class PrintProjects {
    public static void print(List<Project> projects) {
        for (Project project : projects) {
            System.out.print(project.getName().getValue());
            for (ProjectDatasetLink pdl : project.copyDatasetLinks()) {
                Dataset dataset = pdl.getChild();
                System.out.println("  " + dataset.getName().getValue());
            }
        }
    }
}
```

Example: examples/TreeList/PrintProjects.java \(^{85}\)

```c++
#include <omero/client.h>
#include <Usage.h>
#include <AllProjects.h>
#include <PrintProjects.h>
int main(int argc, char* argv[]) {
    std::string host, port, user, pass;
    try {
        host = argv[0];
        port = argv[1];
        user = argv[2];
        pass = argv[3];
    } catch (...) {
        Usage::usage();
    }
    omero::client client(argc, argv);
    int rc = 0;
    try {
        omero::api::ServiceFactoryPrx factory = client.createSession(user, pass);
        std::vector<omero::model::ProjectPtr> projects = AllProjects::getProjects(factory->getQueryService(), user);
        PrintProjects::print(projects);
    } catch (...) {
        client.closeSession();
    }
    return rc;
}
```

Example: examples/TreeList/Main.cpp \(^{86}\)

---

\(^{83}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/TreeList/PrintProjects.m

\(^{84}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/TreeList/PrintProjects.py

\(^{85}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/TreeList/PrintProjects.java

\(^{86}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/TreeList/Main.cpp
function Main(varargin)
try
    host = varargin{1};
    port = varargin{2};
    user = varargin{3};
    pass = varargin{4};
catch ME
    Usage
end
client = omero.client(host, port);
factory = client.createSession(user, pass);
projects = AllProjects(factory.getQueryService(), user);
PrintProjects(projects);
client.closeSession();

Example: examples/TreeList/Main.m

import sys
import omero
import Usage, AllProjects, PrintProjects
if __name__ == "__main__":
    try:
        host = sys.argv[1]
        port = sys.argv[2]
        user = sys.argv[3]
        pass = sys.argv[4]
    except:
        Usage.usage()
    client = omero.client(sys.argv)
    try:
        factory = client.createSession(user, pass)
        projects = AllProjects.getProjects(factory.getQueryService(), user)
        PrintProjects.print_(projects)
    finally:
        client.closeSession()

Example: examples/TreeList/Main.py

import omero.api.ServiceFactoryPrx;
import omero.model.Project;
import java.util.List;
public class Main {
    public static void main(String args[]) throws Exception{
        String host = null, port = null, user = null, pass = null;
        try {
            host = args[0];
            port = args[1];
            user = args[2];
            pass = args[3];
        } catch (Exception e) {
            Usage.usage();
        }
        omero.client client = new omero.client(args);
        try {
            ServiceFactoryPrx factory = client.createSession(user, pass);
            List<Project> projects = AllProjects.getProjects(factory.getQueryService(), user);
            PrintProjects.print_(projects);
        } finally {
            client.closeSession();
        }
}

87 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/TreeList/Main.m
88 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/TreeList/Main.py
16.1.8 Advanced topics

Sudo

If you are familiar with the admin user concept in OMERO, you might wonder if it is possible for administrative users to perform tasks for regular users. Under Unix-based systems this is commonly known as “sudo” functionality. Although not (yet) as straightforward, it is possible to create sessions for other users and carry out actions on their behalf.

```cpp
#include <iostream>
#include <omero/api/IAdmin.h>
#include <omero/api/ISession.h>
#include <omero/client.h>
#include <omero/model/Session.h>

int main(int argc, char* argv[]) {
    Ice::StringSeq args1 = Ice::argsToStringSeq(argc, argv);
    Ice::StringSeq args2(args1); // Copies
    // ticket:1246
    Ice::InitializationData id1;
    id1.properties = Ice::createProperties(args1);
    Ice::InitializationData id2;
    id2.properties = Ice::createProperties(args2);
    omero::client_ptr client = new omero::client(id1);
    omero::client_ptr sudoClient = new omero::client(id2);
    omero::api::ServiceFactoryPrx sf = client->createSession();
    omero::api::ISessionService sessionSvc = sf->getSessionService();
    omero::sys::PrincipalPtr p = new omero::sys::Principal();
    p->name = "root"; // Can change to any user
    p->group = "user";
    p->eventType = "User";
    omero::model::SessionPtr sudoSession = sessionSvc->createSessionWithTimeout( p, 3*60*1000L ); // 3 minutes to live
    std::cout << sudoAdminSvc->getEventContext()->userName;
}
```

Example: examples/OmeroClients/sudo.cpp

```cpp
client = omero.client();
sudoClient = omero.client();
try
    sf = client.createSession('root','ome');
    sessionSvc = sf.getSessionService();
    p = omero.sys.Principal();
    p.name = 'root'; % Can change to any user
    p.group = 'user';
    p.eventType = 'User';
    sudoSession = sessionSvc.createSessionWithTimeout( p, 3*60*1000 ); % 3 minutes to live
    sudoSf = sudoClient.joinSession( sudoSession->getUuid()->getValue() );
    sudoAdminSvc = sudoSf->getAdminService();
    disp(sudoAdmin.Svc.getEventContext().userName);
catch ME
```

89. https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/TreeList/Main.java
90. https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/examples/OmeroClients/sudo.cpp
sudoClient.closeSession();
client.closeSession();
end

Example: examples/OmeroClients/sudo.m

```matlab
import sys
import omero
args = list(sys.argv)
client = omero.client(args)
sudoClient = omero.client(args)
try:
sf = client.createSession("root", "ome")
sessionSvc = sf.getSessionService()
p = omero.sys.Principal()
p.name = "root" # Can change to any user
p.group = "user"
p.eventType = "User"
sudoSession = sessionSvc.createSessionWithTimeout( p, 3*60*1000L ) # 3 minutes to live
sudoSf = sudoClient.joinSession( sudoSession.getUuid().getValue() )
sudoAdminSvc = sudoSf.getAdminService()
print sudoAdminSvc.getEventContext().userName
finally:
sudoClient.closeSession()
client.closeSession()
```

Example: examples/OmeroClients/sudo.py

```python
import java.util.List;
import omero.api.IAdminPrx;
import omero.api.ISessionPrx;
import omero.api.ServiceFactoryPrx;
import omero.model.Session;
import omero.sys.Principal;
public class sudo {
    public static void main(String args[]) throws Exception {
        omero.client client = new omero.client(args);
        omero.client sudoClient = new omero.client(args);
        try {
            ServiceFactoryPrx sf = client.createSession("root", "ome");
            ISessionPrx sessionSvc = sf.getSessionService();
            Principal p = new Principal();
            p.name = "root"; // Can change to any user
            p.group = "user";
            p.eventType = "User";
            Session sudoSession = sessionSvc.createSessionWithTimeout( p, 3*60*1000L ); // 3 minutes to live
            ServiceFactoryPrx sudoSf = sudoClient.joinSession( sudoSession.getUuid().getValue() );
            IAdminPrx sudoAdminSvc = sudoSf.getAdminService();
            System.out.println( sudoAdminSvc.getEventContext().userName );
        } finally {
            sudoClient.closeSession();
client.closeSession();
        }
    }
}
```

Example: examples/OmeroClients/sudo.java

```java
import java.util.List;
import omero.api.IAdminPrx;
import omero.api.ISessionPrx;
import omero.api.ServiceFactoryPrx;
import omero.model.Session;
import omero.sys.Principal;
public class sudo {
    public static void main(String args[]) throws Exception {
        omero.client client = new omero.client(args);
        omero.client sudoClient = new omero.client(args);
        try {
            ServiceFactoryPrx sf = client.createSession("root", "ome");
            ISessionPrx sessionSvc = sf.getSessionService();
            Principal p = new Principal();
            p.name = "root"; // Can change to any user
            p.group = "user";
            p.eventType = "User";
            Session sudoSession = sessionSvc.createSessionWithTimeout( p, 3*60*1000L ); // 3 minutes to live
            ServiceFactoryPrx sudoSf = sudoClient.joinSession( sudoSession.getUuid().getValue() );
            IAdminPrx sudoAdminSvc = sudoSf.getAdminService();
            System.out.println( sudoAdminSvc.getEventContext().userName );
        } finally {
            sudoClient.closeSession();
client.closeSession();
        }
    }
}
```
Proposed

Like the complete examples above, there are several topics which need to be covered in more detail:

• how to detect client/server version mismatches
• how to make asynchronous methods
• how to use client callbacks
• how to make use of your own ObjectFactory

16.1.9 Planned improvements and known issues

Topics to be added

Obviously, this introduction is still not exhaustive by any means. Some topics which we would like to see added here in the near future include:

• more examples of working with the OME-Remote Objects
• examples of all services
• security and ownership
• performance

Code generation

Although not directly relevant to writing a client, it is important to note that much of the code for OMERO Python language bindings, OMERO C++ language bindings, and OMERO Java language bindings is code generated by the BlitzBuild. Therefore, many of the imported and included files in the examples above cannot be found in github94.

We plan to include packages of the generated source code in future releases. Until then, it is possible to find our latest builds on jenkins95 or to build them locally, although some of the generated files are later overwritten by hand-written versions:

• model is located in components/tools/OmeroCpp/src/omero/model/
• OmeroPy is located in components/tools/OmeroPy/src/

Lazy loading and caching

Separate method calls will often return one and the same object, say Dataset#123. Your application, however, will not necessarily recognize them as the same entity unless you explicitly check the id value. A client-side caching mechanism would allow duplicate objects to be handled transparently, and would eventually facilitate lazy loading.

Helper classes

Several types are harder to use than they need be. omero.sys.Parameters, for example, is a class for which native implementations are quite helpful. We have provided omero.sys.ParametersI in all supported languages, and will most likely support more over time:

Other

• Superclasses need to be introduced where possible to replace the ome.model.* interfaces
• Annotation-link-loading can behave strangely if AnnotationLink.child is not loaded.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.
16.2 OMERO Application Programming Interface

All interaction with the OMERO server takes place via several API services available from a ServiceFactory. A service factory is obtained from the client connection e.g. Python:

```python
client = omero.client("localhost")
session = client.createSession("username", "password")  # this is the service factory
adminService = session.getAdminService()  # now we can get/create services
```

- The Service factory API\(^{96}\) has methods for creating Stateless and Stateful services (see below).
  - Stateless services are obtained using “get…” methods e.g. `getQueryService()`
  - Stateful services are obtained using “create…” methods e.g. `createRenderingEngine()`
- Services will provide access to omero.model.objects. You will then need the API for these objects\(^{97}\), e.g. Dataset, Image, Pixels etc.

16.2.1 Services list

The ome.apib\(^{98}\) package in the common component defines the central “verbs” of the OMERO system. All external interactions with the system should happen with these verbs, or services. Each OMERO service belongs to a particular service level with each level calling only on services from lower levels.

Service Level 1 (direct database and Hibernate connections)

- AdminService: src\(^99\), API\(^{100}\) for working with Experimenters, Groups and the current Context (switching groups etc).
- ConfigService: src\(^{101}\), API\(^{102}\) for getting and setting config parameters.
- ContainerService: API\(^{103}\) for loading Project, Dataset and Image hierarchies.
- LdapService: src\(^{104}\), API\(^{105}\) for communicating with LDAP servers.
- MetadataService: API\(^{106}\) for working with Annotations.
- PixelsService: API\(^{107}\) for pixels stats and creating Images with existing or new Pixels.
- ProjectionService API\(^{108}\)
- QueryService: src\(^{109}\), API\(^{110}\) for custom SQL-like queries.
- RenderingSettingsService API\(^{111}\) for copying, pasting & resetting rendering settings.
- RepositoryInfo API\(^{112}\) disk space stats.
- RoiService API\(^{113}\) working with ROIs.
- ScriptService API\(^{114}\) for uploading and launching Python scripts.

---

\(^{96}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/ServiceFactory.html
\(^{97}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/model.html
\(^{98}\) https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/common/src/ome/api
\(^{99}\) https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/IAdmin.java
\(^{100}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/IAdmin.html
\(^{101}\) https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/IConfig.java
\(^{102}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/IConfig.html
\(^{103}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/IContainer.html
\(^{104}\) https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/ILdap.java
\(^{105}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/ILdap.html
\(^{106}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/IMetadata.html
\(^{107}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/IPixels.html
\(^{109}\) https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/IProjection.java
\(^{110}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/IQuery.html
\(^{111}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/IRenderingSettings.html
\(^{112}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/IRepositoryInfo.html
\(^{113}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/IQuery.html
\(^{114}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/IScript.html
• SessionService API\textsuperscript{115} for creating and working with OMERO sessions.
• ShareService API\textsuperscript{116}
• TimelineService API\textsuperscript{117} for queries based on time.
• TypesService API\textsuperscript{118} for Enumerations.
• UpdateService: src\textsuperscript{119}, API\textsuperscript{120} for saving and deleting omero.model objects.

Service Level 2

• IContainer\textsuperscript{121}
• ITypes\textsuperscript{122}

Stateful/Binary Services

• RawFileStore: src\textsuperscript{123}, API\textsuperscript{124}
• RawPixelsStore: src\textsuperscript{125}, API\textsuperscript{126}
• RenderingEngine: src\textsuperscript{127}, API\textsuperscript{128} (see OMERO rendering engine for more)
• ThumbnailStore: src\textsuperscript{129}, API\textsuperscript{130}
• IScale\textsuperscript{131}

A complete list of service APIs can be found here\textsuperscript{132} and some examples of API use in Python are provided. Java or C++ code can use the same API in a very similar manner.

16.2.2 Discussion

Reads and writes

IQuery and IUpdate are the basic building blocks for the rest of the (non-binary) API. IQuery is based on QuerySources and QueryParameters which are explained under Using server queries internally. The goal of this design is to make wildly separate definitions of queries (templates, db-stored, Java code, C# code, …) runnable on the server.

IUpdate takes any graph composed of IObject\textsuperscript{133} objects and checks them for dirtiness. All changes to the graph are stored in the database if the user calling IUpdate has the proper permissions, otherwise an exception is thrown.

Dirty checks follow the Three Commandments:

1. Any IObject-valued field with unloaded set to true is treated as a place holder (proxy) and is re-loaded from the database.
2. Any collection-valued field with a null value is re-loaded from the database.

\textsuperscript{115}http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/ISession.html
\textsuperscript{116}http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/IShare.html
\textsuperscript{117}http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/ITimeline.html
\textsuperscript{118}http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/ITypes.html
\textsuperscript{119}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/IUpdate.java
\textsuperscript{120}http://downloads.openmicroscopy.org/latest/omero/api/ome/api/IUpdate.html
\textsuperscript{121}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/IContainer.java
\textsuperscript{122}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/ITypes.java
\textsuperscript{123}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/RawFileStore.java
\textsuperscript{124}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/RawFileStore.html
\textsuperscript{125}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/RawPixelsStore.java
\textsuperscript{126}http://downloads.openmicroscopy.org/latest/omero/api/ome/api/RawPixelsStore.html
\textsuperscript{127}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/omeis/providers/re/RenderingEngine.java
\textsuperscript{128}http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api/RenderingEngine.html
\textsuperscript{129}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/ThumbnailStore.java
\textsuperscript{130}http://downloads.openmicroscopy.org/latest/omero/api/ome/api/ThumbnailStore.html
\textsuperscript{131}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/IScale.java
\textsuperscript{132}http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/api.html
\textsuperscript{133}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/model/src/ome/model/IObject.java
3. Any collection-valued field with the FILTERED flag is assumed to be dirty and is loaded from the database, with the future option of examining the filtered collection for any new and updated values and applying them to the real collection. (Deletions cannot happen this way since it would be unclear if the object was filtered or deleted.)

Administration

The IAdmin\textsuperscript{134} interface defines all the actions necessary to administer the Server security and firewalls. It is explained further on the OMERO admin interface page.

Model Object Java

Certain operations, like those dealing with data management and viewing, happen more frequently than others (like defining microscopes). Those have been collected in the IContainer\textsuperscript{135} interface. IContainer simplifies a few very common queries, and there is a related package ("omero.gateway.model.*) for working with the returned graphs. OMERO.insight works almost exclusively with the IContainer interface for its non-binary needs.

16.2.3 Examples

```java
// Saving a simple change
Dataset d = iQuery.get(Dataset.class,1L);
d.setName("test");
iUpdate.saveObject(d);

// Creating a new object
Dataset d = new Dataset();
d.setName("test"); // not-null fields must be filled in
iUpdate.saveObject(d);

// Retrieving a graph
Set<Dataset> ds = iQuery.findAllByQuery("from Dataset d left outer join d.images where d.name = 'test'");
```

16.2.4 Stateless versus stateful services

A stateless service has no client-noticeable lifecycle and all instances can be treated equally. A new stateful service, on the other hand, will be created for each client-side proxy (see the ServiceFactory.create\textsuperscript{*} methods). Once obtained, a stateful service proxy can only be used by a single user. After task completion, the service should be closed (proxy.close()) to free up server resources.

16.2.5 How to write a service

A tutorial is available at How To create a service. In general, if a properly annotated service is placed in any JAR of the OMERO EAR file (see Build System for more) then the service will be deployed to the server. In the case of OMERO.blitz, the service must be properly defined under components/blitz/resources\textsuperscript{136}.

16.2.6 OMERO annotations for validation

The server-side implementation of these interfaces makes use of ((JDK5)) Structured annotations and an AOP interceptor to validate all method parameters. Calls to pojos.findContainerHierarches are first caught by a method interceptor, which checks for annotations on the parameters and, if available, performs the necessary checks. The interceptor also makes proactive checks. For a range of parameter types (such as Java Collections) it requires that annotations exist and will refuse to proceed if not implemented.

\textsuperscript{134}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/IAdmin.java
\textsuperscript{135}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/IContainer.java
\textsuperscript{136}https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/blitz/resources
An API call of the form:

```java
pojos.findContainerHierarches(Class, Set, Map)
```

is implemented as

```java
pojos.findContainerHierarchies(@NotNull Class, @NotNull @Validate(Integer.class) Set, Map)
```

See also:

* Using server queries internally, OMERO rendering engine, Exception handling

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

## 16.3 OMERO admin interface

The one central interface for administering the OMERO security system is IAdmin. Though several of the methods are restricted to system users (root and other administrators), many are also for general use. The RolesAllowed annotations on the LocalAdmin\(^{137}\) class define who can use which methods.

### 16.3.1 Actions available through IAdmin and IUpdate

A couple of the methods in the IAdmin interface are also available implicitly through IUpdate, the main interface for updating the database. This duplication is mainly useful for large scale changes, such as changing the permissions to an entire object graph.

- changePermissions
- changeGroup

The following shows how these methods can be equivalently used:

```java
// setup
ServiceFactory sf = new ServiceFactory();
IAdmin iAdmin = sf.getAdminService();
IUpdate iUpdate = sf.getUpdateService();
Image myImg = ...; //

// using IAdmin -- let’s change the group of myImg
// and then make it group private.
iAdmin.changeGroup(myImg, new ExperimenterGroup( 3L, false ));
iAdmin.changePermissions( myImg, new Permissions( Permissions.GROUP_PRIVATE ));

// and do the same using Details and IUpdate
myImg.getDetails().setPermissions( new Permissions( Permissions.GROUP_PRIVATE ));
myImg.getDetails().setGroup( new ExperimenterGroup( 3L, false ));
iUpdate.saveObject( myImg );
```

The benefit of the second method is the batching of changes into a single call. The benefit of the first is at most explicitness. Note, however, that changing any of the values of Details which are not also changeable through IAdmin will result in a SecurityViolation.

### 16.3.2 Actions only available through IAdmin

The rest of the write methods provided by IAdmin are disallowed for IUpdate and will throw SecurityViolations. This includes adding users, groups, user/group maps, events, enums, or similar. ( Enums here are a special case, because they are created

---

\(^{137}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/api/local/LocalAdmin.java
not through IAdmin but through ITypes). A system administrator may be able to use IUUpdate to create these “System-Types” but using IAdmin is safer, cleaner, and guaranteed to work in the future.

The password methods and synchronizeLoginCache are also special cases in that they have no equivalent in any other API.

### 16.3.3 Similarities between IAdmin and IQuery

All of the read methods provided by IAdmin are also available from IQuery, that is, the IAdmin (currently) provide no special context or security privileges. However, having all of the methods in one interface reduces code duplication, which is especially useful when you want the entire user/group graph as provided by getExperimenter/getGroup/lookupExperimenter/lookupGroup.

See also:

*OMERO Application Programming Interface*

---

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

---

### 16.4 Deleting in OMERO

Deleting data in OMERO is complex due to the highly linked nature of data in the database. For example, an Image has links to Datasets, Comments, Tags, Instrument, Acquisition metadata etc. If the image is deleted, some of this other data should remain and some should be deleted with the image (since it has no other relevance).

In the 4.2.1 release of OMERO, an improved deleting service was introduced to fix several problems or requirements related to the delete functionality (see #2615138 for tickets):

- Need a better way to define what gets deleted when certain data gets deleted (e.g. Image case above)
- Need to be able to configure this definition, since different users have different needs
- Deleting large amounts of data (e.g. Plate of HCS data) was too memory-intensive (data was loaded from the database during delete)
- Poor logging of deletes
- Large deletes (e.g. screen data) take time: Clients need to be able to keep working while deletes run ‘in the background’
- Binary data (pixels, thumbnails, files etc) was not removed at delete time - required sysadmin to clean up later

Future releases will continue this work (see #2911139) and the 5.1.0 release of OMERO offers a new implementation of deletion.

#### 16.4.1 Finality of deletion

Import in OMERO 5.x uploads the image and companion files intact and stores them within subdirectories of the directory configured by the value of `omero.managed.dir`, typically ManagedRepository. The files relating to a specific fileset are stored together on the server’s filesystem and they are read by Bio-Formats when images are viewed in OMERO clients. If any of a fileset’s files, or the corresponding entries for them in the database, are deleted, then the fileset may no longer be readable. If all the fileset’s files are deleted, the fileset will certainly be unreadable, and there is no ‘undo’ that will bring it back.

#### 16.4.2 Delete behavior (technical)

Configuring what gets deleted is done using XML files. Since OMERO 5.1, the delete behavior defaults to a *Model graph operations* implementation that is configured by components/blitz/resources/ome/services/blitz-graph-rules.xml140.

138[https://trac.openmicroscopy.org/ome/ticket/2615](https://trac.openmicroscopy.org/ome/ticket/2615)
139[https://trac.openmicroscopy.org/ome/ticket/2911](https://trac.openmicroscopy.org/ome/ticket/2911)
140[https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/resources/ome/services/blitz-graph-rules.xml](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/resources/ome/services/blitz-graph-rules.xml)
Delete Image

The general delete behavior for deleting an Image is to remove every piece of data from the database that was added when the image was imported, removing pixel data and thumbnails from disk. In addition, the following data is deleted:

- Comments on the image
- Rating of the image
- ROIs for this image (see below)
- Image Rendering settings for yourself and other users

Optional - In OMERO.web and OMERO.insight, you will be asked whether you also want to delete:

- Files attached to the image (if not linked elsewhere). In that case, the binary data will be removed from disk too.
- Your own tags on the image (if not used elsewhere)

The same option is available when deleting dataset, project, plate, screen.

Delete Dataset or Project

When deleting a Project or Dataset, you have the option to also delete tags and annotations (as for Image above). You also can choose whether to ‘delete contents’. This will delete any Datasets (or Images) that are contained in the Project (or Dataset).

However, Datasets and Images will not get deleted if they are also contained in other Projects or Datasets respectively.

If a user decides to delete/keep the annotations (see Optional above) when deleting a Project (or Dataset) and its contents, the rule associated to the annotation will be apply to all objects.

Delete Screen, Plate or Plate Acquisition

When deleting a Screen, you have the option to also delete tags and annotations. You also can choose whether to ‘delete contents’. This will delete any Plates that are contained in the Screen.

When deleting a Plate, you have the option to also delete tags and annotations but NOT the option to ‘delete contents’.

If the Plate has Plate Acquisitions, you can delete one or more Plate Acquisition at once.

Delete Tag/Attachment

You can delete a Tag/Attachment, and it will be removed from all images. However you cannot delete a Tag/Attachment if it has been used by another user in the same collaborative group. This is to prevent potential loss of significant amount of annotation effort by other users. You will need to get the other users to first remove your Tag/Attachment where they have used it, before you can delete it.

Known Issue: if the owner of the Tag/Attachment is also an owner of the group (e.g. PI), they will be able to delete their Tag/Attachment, even if others have used it.

Delete multi-file Images and Image sets

An Image, or a set of Images, may come from a single file or a set of dependent files. For instance, a single Leica LIF file may contain many Images, as may a Zeiss mdb file with .ism files. On the other hand, some file formats, like Deltavision with log file, or the original ICS file format, use multiple files to represent a single Image. At import time, these groups of related files and Images are organized into Filesets: a Fileset is a set of files that encode a set of Images. The simplest case where there is one file per Image still has a corresponding Fileset.

Even if many Images come from the same file, they may be separately selected and viewed in client software. However, at least at present, a Fileset may not be partially deleted: either all the files and Images from it are deleted, or none are. So, for instance, the Images from the same Leica LIF file may be deleted only all at once, and the Deltavision log file is not deleted separately from the main file. The same applies to high-content screening data: a Plate with its Wells and Images are all stored in one Fileset and may be deleted only together.
Each Fileset has a corresponding directory on the server in which, perhaps in subdirectories, all its files are stored. All the file paths for an Image’s Fileset can be accessed from the tool-bar at the top of the right-hand panel.

**Delete in collaborative group**

Some more discussion of delete issues in a collaborative group, where your data are linked to data of other users, can be found on the *Groups and permissions system* page.

- A user cannot remove Images from another user’s Dataset, or remove Datasets (or Plates) from Projects (or Screens).
- A user cannot delete anything that belongs to another user.

**Group owner rights**

An owner of the group, usually a PI, can delete anything that belongs to other members of the group.

**Edge cases**

These are ‘known issues’ that may cause problems for some users (not for most). These will be resolved in future depending on priority.

- Other users’ ROIs (and associated measurements) are deleted from images.
- Multiply-linked objects are unlinked and not deleted e.g. Project p1 contains two Datasets d1 and d2, Project p2 contains Dataset d1. If the Project p1 is deleted, the Dataset d1 is only unlinked from p1 and not completely deleted.

**Binary data**

When Images, Plates or File Annotations have been successfully deleted from the database the corresponding binary data is deleted from the binary repository (see *Unix* and *Windows* versions). It is possible that some files may not be successfully deleted if they are locked for any reason. This is a known problem on Windows servers. In this case, the undeleted files can be removed manually via `omero admin cleanse`. This also deletes any empty directories left behind after the binary data that they contained has been deleted.

**Warning:** Do not run `cleanse` as an operating system user while logged into OMERO as a non-administrative user as this will lead to data loss. Instead, you should always run as an administrative user such as “root”. See this [announcement](http://www.openmicroscopy.org/community/viewtopic.php?f=11&t=8060) for further details.

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 16.5 OMERO Import Library

The Import Library is a re-usable framework for building import clients. Several are provided by the OMERO team directly:

- the integrated *importer*
- *Command Line Importer* tool

#### 16.5.1 Components

The primary classes which make up the Import Library are:

- `ImportLibrary.java`® itself, which is the main driver

®[https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/src/omeformats/importer/ImportLibrary.java](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/src/omeformats/importer/ImportLibrary.java)
• ImportCandidates.java\textsuperscript{142} which takes file paths and determines the proper files to import
• ImportConfig.java\textsuperscript{143}, an extensible mechanism for storing the properties used during import
• ImportEvent.java\textsuperscript{144}, the various events raised during import to IObserver and IObserverable implementations
• OMEROMetadataStoreClient.java\textsuperscript{145}, the low-level connection to the server
• OMEROWrapper.java\textsuperscript{146}, the OMERO adapter for the Bio-Formats ImageReaders class
• In OMERO.insight, the main entry point is the importImage method of OMEROGateway.java\textsuperscript{147}
• In the CLI, the main entry point is the CommandLineImporter\textsuperscript{148} class

16.5.2 Earlier Import Workflow

Prior to OMERO 5.0, the import workflow was very much client-side. Using the ImportLibrary a client would determine the import candidates and then import the image. The import phase would comprise copying the pixel data to the OMERO data directory, writing the metadata into the database, and optionally copying the original file to the OMERO data directory for archiving.

16.5.3 FS Managed Repository Import Workflow

From 5.0 the workflow has changed. The client still determines the import candidates but the client-side import process simply uploads the original files to the OMERO data directory and then uses the ManagedRepository service to initiate a server-side import. On the server the import is then completed by writing the metadata into the database. After import, pixel data is accessed directly from the original files using Bio-Formats. This means that data files are no longer duplicated and any nested directory structure is preserved. It also allows OMERO to take advantage of pre-generated pyramids available in some formats e.g. SVS.

For full details of the import workflow see Import under OMERO.fs.

16.5.4 Example

The CommandLineImporter.java class shows a straightforward import. An ErrorHandler instance is passed both to the ImportCandidates constructor (since errors can occur while parsing a directory) and to the ImportLibrary. This and other handlers receive ImportEvents which notify listeners of the state of the current import.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

16.6 TempFileManager

Class to be used by Working with OMERO and server components to allow a uniform creation of temporary files and folders with a best-effort guarantee of deleting the resources on exit. The manager searches three locations in order, taking the first which allows lockable write-access (See #1653\textsuperscript{149}):

• The environment property setting OMERO_TMPDIR
• The user’s home directory, for example specified in Java via System.getProperty("user.home")
• The system temp directory, in Java System.getProperty("java.io.tmpdir") and in Python tempfile.gettempdir()

\textsuperscript{142}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/src/ome/formats/importer/ImportCandidates.java
\textsuperscript{143}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/src/ome/formats/importer/ImportConfig.java
\textsuperscript{144}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/src/ome/formats/importer/ImportEvent.java
\textsuperscript{145}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/src/ome/formats/importer/OMEROMetadataStoreClient.java
\textsuperscript{146}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/src/ome/formats/importer/OMEROWrapper.java
\textsuperscript{147}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/insight/SRC/org/openmicroscopy/shoola/env/data/OMEROGateway.java
\textsuperscript{148}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/src/ome/formats/importer/cli/CommandLineImporter.java
\textsuperscript{149}https://trac.openmicroscopy.org/ome/ticket/1653
16.6.1 Creating temporary files

For the user "ralph",

```python
from omero.util.temp_files import create_path
path = create_path("omero",".tmp")
```

or

```python
import omero.util.TempFileManager
File file = TempFileManager.create_path("omero",".tmp")
```

both produce a file under the directory:

```
/tmp/omero_ralph/SPID/omero$RANDOM.tmp
```

where SPID is the current process id and $RANDOM is some random sequence of alphanumeric characters.

16.6.2 Removing files

If `remove_path` is called on the return value of `create_path`, then the temporary resources will be cleaned up immediately. Otherwise, when the Java or Python process exits, they will be deleted. This is achieved in Java through `Runtime#addShutdownHook(Thread)` and in Python via `atexit.register()`.

16.6.3 Creating directories

If an entire directory with a unique directory is needed, pass "true" as the “folder” argument of the `create_path` method:

```python
create_path("omero", ".tmp", folder = True)
```

and

```python
TempFileManager.create_path("omero", ".tmp", true);
```

**Note:** All contents of the generated directory will be deleted.

**See also:**

#1534150

---

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

16.7 Exception handling

16.7.1 Client exceptions

The exceptions which can be received by a client due to a remote call on the OMERO server are all defined in `components/blitz/resources/omero/ServerErrors.ice` (included below). This file contains two separate hierarchies rooted at `Ice::Exception` and `omero::ServerError`.

---

150 https://trac.openmicroscopy.org/ome/ticket/1534
151 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/resources/omero/ServerErrors.ice
For a better understanding of how to handle exceptions, please read both of the *.ice files carefully, and see Working with OMERO for examples of exception handling.

```c++
/*
 * $Id$
 *
 * Copyright 2007 Glencoe Software, Inc. All rights reserved.
 * Use is subject to license terms supplied in LICENSE.txt
 *
 */
 ifndef OMERO_SERVERERRORS_ICE
 define OMERO_SERVERERRORS_ICE
 include <Glacier2/Session.ice>
 /**
 * Exceptions thrown by OMERO server components. Exceptions thrown client side
 * are available defined in each language binding separately, but will usually
 * subclass from "ClientError"
 *
 * including examples of what a appropriate try/catch block would look like.
 *
 */
 <pre>
 * All exceptions that are thrown by a remote call (any call on a *Prx instance)
 * will be either a subclass of [Ice::UserException] or [Ice::LocalException].
 * @a href="http://doc.zeroc.com/display/Ice/Run-Time+Exceptions#Run-TimeExceptions-InheritanceHiearchi" from the Ice manual shows the entire exception hierarchy. The exceptions described in
 * this file will subclass from [Ice::UserException]. Other Ice-runtime exceptions subclass
 * from [Ice::LocalException].
 *
 </pre>
 *
 OMERO Specific:
 ---------------
 ServerError (root server exception)
 |
 | InternalException (server bug)
 |
 | ResourceError (non-recoverable)
 | NoProcessorAvailable
 |
 | ConcurrencyException (recoverable)
 | ConcurrentModification (data was changed)
 | OptimisticLockException (changed data conflicts)
 | LockTimeout (took too long to acquire lock)
 | TryAgain (some processing required before server is ready)
 |
 TooManyUsersException
 |
 DatabaseBusyException
 |
 ApiException (misuse of services)
 |
 OverUsageException (too much)
 |
 QueryException (bad query string)
 |
 ValidationException (bad data)
 |
 SecurityViolation (some no-no)
 |
 GroupSecurityViolation
 |
 PermissionMismatchGroupSecurityViolation
 |
 ReadOnlyGroupSecurityViolation
 |
 SessionException
 |
 RemovedSessionException (accessing a non-extant session)
 |
 SessionTimeoutException (session timed out; not yet removed)
 |
 ShutdownInProgress (session on this server will most likely be destroyed)
 */
```

16.7. Exception handling
However, in addition to \[\text{Ice::LocalException}\] subclasses, the \text{Ice} runtime also defines subclasses of \[\text{Ice::UserException}\]. In some cases, OMERO subclasses from these exceptions. The subclasses shown below are not exhaustive, but show those which an application’s exception handler may want to deal with.

\[
\begin{array}{c}
\text{Ice::Exception (root of all Ice exceptions)} \\
\text{ } |
\end{array}
\]

\[
\begin{array}{c}
\text{ | \text{Ice::UserException (super class of all application exceptions)}} \\
\text{ | | \text{Glacier2::CannotCreateSessionException (1 of 2 exceptions throwable by createSession)}} \\
\text{ | | | \text{omerod::AuthenticationException (bad login)}} \\
\text{ | | | \text{omerod::ExpiredCredentialException (old password)}} \\
\text{ | | | \text{omerod::WrappedCreateSessionException (any other server error during createSession)}} \\
\text{ | | | \text{omerod::licenses::NoAvailableLicensesException (see tools/licenses/resources/omero/LicensesAPI.ice)}} \\
\text{ | | \text{Glacier2::PermissionDeniedException (other of 2 exceptions throwable by createSession)}} \\
\text{ | \text{Ice::LocalException (should generally be considered fatal. See exceptions below)}} \\
\text{ | | \text{Ice::ProtocolException (something went wrong on the wire. Wrong version?)}} \\
\text{ | | \text{Ice::RequestFailedException}} \\
\text{ | | \text{ObjectNotExistException (Service timeout or similar?)}} \\
\text{ | | \text{OperationNotExistException (Improper use of uncheckedCast?)}} \\
\text{ | | \text{Ice::UnknownException (server threw an unexpected exception. Bug!)}} \\
\text{ | | \text{Ice::TimeoutException}} \\
\text{ | \text{Ice::ConnectTimeoutException ( Couldn’t establish a connection. Retry?)}}
\end{array}
\]

\[
\begin{array}{c}
\text{module omero}
\end{array}
\]

\[
\begin{array}{c}
\text{*/}
\end{array}
\]

\[
\begin{array}{c}
\text{exception ServerError}
\end{array}
\]

\[
\begin{array}{c}
\text{ }
\end{array}
\]

\[
\begin{array}{c}
\text{string serverStackTrace;}
\end{array}
\]

\[
\begin{array}{c}
\text{string serverExceptionClass;}
\end{array}
\]

\[
\begin{array}{c}
\text{string message;}
\end{array}
\]

\[
\begin{array}{c}
\text{// SESSION EXCEPTIONS -------------------------------}
\end{array}
\]

\[
\begin{array}{c}
\text{/**}
\end{array}
\]

\[
\begin{array}{c}
\text{Base session exception, though in the OMERO.blitz implementation, all exceptions thrown}
\text{by the Glacier2 must subclass CannotCreateSessionException. See below.}
\end{array}
\]

\[
\begin{array}{c}
\text{exception SessionException extends ServerError}
\end{array}
\]

\[
\begin{array}{c}
\text{}
\end{array}
\]

\[
\begin{array}{c}
\text{/**}
\end{array}
\]

\[
\begin{array}{c}
\text{Session has been removed. Either it was closed, or it timed out and one "SessionTimeoutException" has already}
\end{array}
\]
* been thrown.
*/
	exception RemovedSessionException extends SessionException
	{
	    
	/**
	 * Session has timed out and will be removed.
	 */
	exception SessionTimeoutException extends SessionException
	{
	    
	/**
	 * Server is in the progress of shutting down which will
	 * typically lead to the current session being closed.
	 */
	exception ShutdownInProgress extends SessionException
	{
	    
	/**
	 * createSession() is a two-phase process. First, a PermissionsVerifier is
	 * called which must return true; then a SessionManager is called to create
	 * the session (ServiceFactory). If the PermissionsVerifier returns false,
	 * then PermissionDeniedException will be thrown. This, however, cannot be
	 * subclassed and so string parsing must be used.
	 */
	/**
	 * Thrown when the information provided omero.createSession() or more
	 * specifically Glacier2.RouterPrx.createSession() is incorrect. This
	 * does -not- subclass from the omero.ServerError class because the
	 * Ice Glacier2::SessionManager interface can only throw CCSEs.
	 */
	exception AuthenticationException extends Glacier2::CannotCreateSessionException
	{
	    
	/**
	 * Thrown when the password for a user has expired. Use: ISession.changeExpiredCredentials()
	 * and login as guest. This does -not- subclass from the omero.ServerError class because the
	 * Ice Glacier2::SessionManager interface can only throw CCSEs.
	 */
	exception ExpiredCredentialException extends Glacier2::CannotCreateSessionException
	{
	    
	/**
	 * Thrown when any other server exception causes the session creation to fail.
	 * Since working with the static information of Ice exceptions is not as easy
	 * as with classes, here we use booleans to represent what has gone wrong.
	 */
	exception WrappedCreateSessionException extends Glacier2::CannotCreateSessionException
	{
	    bool concurrency;
	    long backOff;  /* Only used if ConcurrencyException */
	    string type;   /* Ice static type information */
	}


}// RESOURCE
/**
 * Unrecoverable error. The resource being accessed is not available.
exception ResourceError extends ServerError
{
};
/**
* A script cannot be executed because no matching processor
* was found.
*/
exception NoProcessorAvailable extends ResourceError
{

/**
 * Number of processors that responded to the inquiry.
 * If 1 or more, then the given script was not acceptable
 * (e.g. non-official) and a specialized processor may need
 * to be started.
 **/
 int processorCount;
};
// CONCURRENCY
/**
* Recoverable error caused by simultaneous access of some form.
*/
exception ConcurrencyException extends ServerError
{
 long backOff; /* Backoff in milliseconds */
};
/**
* Currently unused.
*/
exception ConcurrentModification extends ConcurrencyException
{
};
/**
* Too many simultaneous database users. This implies that a
* connection to the database could not be acquired, no data
* was saved or modified. Clients may want to wait the given
* backOff period, and retry.
*/
exception DatabaseBusyException extends ConcurrencyException
{
};
/**
* Conflicting changes to the same piece of data.
*/
exception OptimisticLockException extends ConcurrencyException
{
};
/**
* Lock cannot be acquired and has timed out.
*/
exception LockTimeout extends ConcurrencyException
{
 int seconds; /* Informational field on how long timeout was */
};
/**
* Background processing needed before server is ready
*/
exception TryAgain extends ConcurrencyException
{
};
exception MissingPyramidException extends ConcurrencyException
{
 long pixelsID;
};
// API USAGE
exception ApiUsageException extends ServerError
{
};
exception OverUsageException extends ApiUsageException
{
/**
 *
 */
exception QueryException extends ApiUsageException
{
};
exception ValidationException extends ApiUsageException
{
};
// SECURITY
exception SecurityViolation extends ServerError
{
};
exception GroupSecurityViolation extends SecurityViolation
{
};
exception PermissionMismatchGroupSecurityViolation extends SecurityViolation
{
};
exception ReadOnlyGroupSecurityViolation extends SecurityViolation
{
};
// OMEROFS
/**
 * OmeroFSError
 *
 * Just one catch-all UserException for the present. It could be
 * subclassed to provide a finer grained level if necessary.
 * It should be fitted into or subsumed within the above hierarchy
 **/
exception OmeroFSError extends ServerError
{
    string reason;
};
#endif // OMERO_SERVERERRORS_ICE

16.7.2 Server exceptions

Due to the strict API boundary enforced by Ice, the client and server exception hierarchies, though related, are distinct. The discussion below is possibly of interest for server developers only. Client developers should refer to the information and examples under Working with OMERO.

Interceptor

Exception handling in the OMERO is centralized in an Aspect-oriented programming interceptor (source code\(^\text{152}\)). All exceptions thrown by code are caught in a try {} catch (Throwable t) {} block. Exceptions which do not subclass ome.conditions.RootException\(^\text{153}\) are wrapped in an ome.conditions.InternalException\(^\text{154}\).

\(^{152}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/services/util/ServiceHandler.java

\(^{153}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/model/src/ome/conditions/RootException.java

\(^{154}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/model/src/ome/conditions/InternalException.java
The only exceptions to this are any interceptors which may be run before the exception handler is run. The order of interceptors is defined in services.xml

Hierarchy

The current exception hierarchy (package ome.conditions) used is as follows:

- RootException
  - InternalException - should not reach the client; Bug! Contact administrator e.g. NullPointerException, assertion failed, etc.
  - ResourceError - fatal error in server, e.g. OutOfMemory, disk space full, the database is in illegal state, etc.
  - DataAccessException
    * SecurityViolation - do not do that! E.g. edit locked project, create new user.
    * OptimisticLockException - re-load and compare e.g. “someone else has already updated this project”
    * ApiUsageException - something wrong with how you did things e.g. IllegalStateException, object uninitialized, etc.
    * ValidationException - something wrong with what you sent; sends list of fields, etc.; edit and retry, e.g. no ”?” in image names.

where the colors indicate:

Abstract
FixAndRetryConditions
RetryConditions
NoRecourseConditions

Any other exception which reaches the client should be considered an OutOfServiceException, meaning that something is (hopefully only) temporarily wrong with the server, e.g. no connection, server down, server restarting. But since this cannot be caught since the server cannot be reached, there is no way to guarantee that a real OutOfServiceException is thrown.

16.7.3 Moving forward

FixAndRetryConditions need to have information about what should be fixed, like a Validation object which lists fields with error messages. A RetryCondition could have a back-off value to prevent too frequent retries.

Questions

- What data should be available in the exceptions?
- What other logic do we want on our exceptions, keeping in mind they will have to be re-implemented in all target languages?

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

16.8 Omero logging

All OMERO components written in Java use the SLF4J logging facade, typically backed by Logback; all components written in python use the built-in logging module.

155 [https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/resources/ome/services/services.xml](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/resources/ome/services/services.xml)
156 [https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/model/src/ome/conditions](https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/model/src/ome/conditions)
158 [http://logback.qos.ch/](http://logback.qos.ch/)
16.8.1 Java clients

Java clients log to $HOME/omero/log. The number of files and their size are limited.

logback-cli.xml controls the output for the command line importer: all logging goes to standard err, while useful output (pixel ids, or used files) goes to standard out. It is contained within the blitz.jar itself. Therefore, to modify the settings use -Dlogback.configurationFile=/path/to/logback.xml or similar.

OMERO.insight logging is configured via logback.xml which is available in the config/ directory of any OMERO.insight install.

16.8.2 Java servers

Java server components are configured by passing -Dlogback.configurationFile=etc/logback.xml to each Java process. Entry.java guarantees that the logback.xml file is read periodically so that changes to your logging configuration do not require a restart.

By default, the output from logback is sent to: var/log/<servername>.log. Once files reach a size of 500MB, they are rolled over to <servername>.log.1, <servername>.log.2, etc. Once the files have rolled over, you can safely delete or compress (bzip2, gzip, zip) them. Alternatively, once you are comfortable with the stability of your server, you can either reduce logging or the number and size of the files kept. Note: if something goes wrong with your server installation, the log files can be very useful in tracking down issues.

In addition, each import process logs to a file under the managed repository which matches the timestamped fileset directory’s name. For example, if an imported fileset is uploaded to /OMERO/ManagedRepository/userA_1/2013-06/17/12-00-00-00.000, then the log file can be found under /OMERO/ManagedRepository/userA_1/2013-06/17/12-00-00.000.log.

16.8.3 Python servers

Python servers are configured by a call to omero.util.configure_server_logging(props). The property values are taken from the configuration file passed to the server via icegridnode. For example, the config file for Processor-0 can be found in var/master/servers/Processor-0/config/config. These values come from etc/grid/templates.xml.

All the “omero.logging.” properties can be overwritten in your etc/grid/default.xml file (or on Windows, etc/grid/windefault.xml). See the “Profile” properties block for how to configure for your site.

Similar to logback, logging is configured to be written to var/log/<servername>.log and to maintain 9 backups of at most 500MB.

16.8.4 stdout and stderr

Though all components try to avoid it, some output will still go to stdout/stderr. On non-Windows systems, all of this output will be sent to the var/log/master.out and var/log/master.err files.

16.8.5 Windows stdout and stderr

On Windows, the state of stdout and stderr is somewhat different. No information will be written to master.out, master.err, or similar files. Instead, what logging is produced will go to the Windows Event Viewer, but finding error situations can be considerably more challenging (See #1449 for more information).

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of security or major stability problems.
OMERO Documentation, Release 5.2.6

of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

16.9 Using the new 5.1 graph requests

16.9.1 Migration is required

OMERO 5.1.0 introduced a new implementation of Model graph operations offered through the API via Chgrp2\(^{163}\), Chown2\(^{164}\), Delete2\(^{165}\), and their superclass GraphModify2\(^{166}\). OMERO 5.1.2 added Chmod2\(^{167}\). The legacy request operations Chgrp\(^{168}\), Chmod\(^{169}\), Chown\(^{170}\), Delete\(^{171}\), and their superclass GraphModify\(^{172}\), are now deprecated and will be removed in OMERO 5.3. Now is the time to adjust client code accordingly so that the OME team can fix any regressions before the release of OMERO 5.3.

16.9.2 Target objects

For specifying which model objects to operate on, instead of using one request for each object through GraphModify\(^{173}\)’s type and id data members, use GraphModify2\(^{174}\)’s targetObjects which allows specification of multiple model object classes, each with an unordered list of IDs, all in a single request. To specify a type, no longer use /-delimited paths, but instead just the class name, e.g. Image instead of /Image. To achieve a root-anchored subgraph operation use SkipHead\(^{175}\) to wrap your request: for instance, for /Image/Pixels/RenderingDef, set the SkipHead request’s targetObjects to the image(s), and set startFrom to RenderingDef.

16.9.3 Translating options

GraphModify\(^{176}\)’s options data member has its related analog in GraphModify2\(^{177}\)’s childOptions, an ordered list of ChildOption\(^{178}\) instances, each of which allows its applicability to annotations to be limited by namespace. Some examples:

- To move a dataset with all its images, removing those images from other datasets where necessary, use Chgrp2 with a ChildOption’s includeType set to Image.
- To delete a dataset without deleting any images at all from it, use Delete2 with a ChildOption’s excludeType set to Image.
- To delete annotations except for the tags that are in a specific namespace, use Delete2 with a ChildOption’s excludeType set to TagAnnotation and includeNs set to that namespace.

GraphUtil.java\(^{179}\) includes the translateOptions method that may give additional insight on how to translate the previous style of option. This method too will be removed in OMERO 5.3.

---

\(^{163}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Chgrp2.html

\(^{164}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Chown2.html

\(^{165}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Delete2.html

\(^{166}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/GraphModify2.html

\(^{167}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Chmod2.html

\(^{168}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Chgrp.html

\(^{169}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Chmod.html


\(^{171}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Chown.html

\(^{172}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Delete.html

\(^{173}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/GraphModify.html

\(^{174}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/GraphModify2.html

\(^{175}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/SkipHead.html

\(^{176}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/GraphModify.html

\(^{177}\) http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/GraphModify2.html


\(^{179}\) https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/src/omero/cmd/graphs/GraphUtil.java
16.9.4 Examples in Python

Move images

```python
chgrps = DoAll()
chgrps.requests = [Chgrp(type="/Image", id=n, grp=5) for n in [1,2,3]]
sf.submit(chgrps)
```

becomes,

```python
chgrp = Chgrp2(targetObjects={'Image': [1,2,3]}, groupId=5)
sf.submit(chgrp)
```

Delete plate, but not annotations

```python
keepAnn ={"/Annotation": "KEEP"}
delete = Delete(type="/Plate", id=8, options=keepAnn)
sf.submit(delete)
```

becomes,

```python
keepAnn = [ChildOption(excludeType=['Annotation'])]
delete = Delete2(targetObjects={'Plate': [8]}, childOptions=keepAnn)
sf.submit(delete)
```

Delete an image’s rendering settings

```python
delete = Delete(type="/Image/Pixels/RenderingDef", id=6)
sf.submit(delete)
```

becomes,

```python
anchor = {'Image': [6]}
targets = ['RenderingDef']
delete = SkipHead(targetObjects=anchor, startFrom=targets,
request=Delete2())
sf.submit(delete)
```

16.9.5 Java request factory

A utility class Requests.java\(^{100}\) provides convenient instantiation of graph requests. This class allows the requests from the above Python examples to be created by,

```java
// move images
Chgrp2 example1 = Requests.chgrp("Image", Arrays.asList(1L,2L,3L), 5L);

// delete plate, but not annotations
ChildOption childOption = Requests.option(null, "Annotation");
```

\(^{100}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/src/omero/gateway/util/Requests.java
Delete2 example2 = Requests.delete("Plate", 8L, childOption);

// delete an image's rendering settings
SkipHead example3 =
    Requests.skipHead("Image", 6L, "RenderingDef", new Delete2());

16.9. Using the new 5.1 graph requests
Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

17.1 OME-Remote Objects

OMERO is based on the OME data model which can appear overly complex for new users. However, the core entities you need for getting started are much simpler.

Images in OMERO are organized into a many-to-many container hierarchy: “Project” -> “Dataset” -> “Image”. These containers (and various other objects) can be annotated to link various types of data. Annotation types include Comment (string), Tag (short string), Boolean, Long, Xml, File attachment etc.

Images are represented as Pixels with 5 dimensions: X, Y, Z, Channel, Time.

At the core of the work on the Open Microscopy Environment\(^1\) is the definition of a vocabulary for working with microscopic data. This vocabulary has a representation in the XML specification\(^2\), in the database (the data model), and in code. This last representation is the object model with which we will concern ourselves here.

\(^1\)http://www.openmicroscopy.org/site
\(^2\)http://www.openmicroscopy.org/site/support/ome-model/ome-xml/
Because of its complexity, the object model is generated from a central definition\(^3\) using our own code-generator\(^4\). It relies on no libraries and can be used in both the server and the RMI clients. The relationships among the objects are enumerated in a cross-referenced reference document. OMERO.blitz uses a second mapping\(^5\) to generate OMERO Java language bindings, OMERO Python language bindings, and OMERO C++ language bindings classes, which can be mapped\(^6\) back and forth to the server object model. This document discusses only the server object-model and how it is used internally.

Instances of the object model have no direct interaction with the database, rather the mapping is handled externally by the O/R framework, Hibernate\(^7\). That means, by and large, generated classes are data objects, composed only of getter and setter fields for fields representing columns in the database, and contain no business logic. However, to make working with the model easier, and perhaps more powerful, there are several features which we have built in.

**Note:** The discussion here of object types is still relevant but uses the ome.model.* objects for examples. These are server internal types which may lead to some confusion. Clients work with omero.model.* objects. This documentation will eventually be updated to reflect both hierarchies.

### 17.1.1 OMERO type language

The Model has two general parts: first, the long-studied and well-established core model and second, the user-specified portion. It is vital that there is a central definition of both parts of the object model. To allow users to easily define new types, we need a simple domain specific language (or little language) which can be mapped to Hibernate mapping files. See an example at:

* components/model/resources/mappings/acquisition.ome.xml\(^8\)

From this DSL, various artifacts can be generated: XML Schema, Java classes, SQL for generating tables, etc. The ultimate goal is to have no exceptions in the model.

![Figure 1](image)

Conceptually, the XSD files under the components/specification source directory are the starting point for all code generation. Currently however, the files under components/model/resources/mappings\(^9\) are hand-written based on the XSD files.

The ant task created from the components/dsl/src\(^10\) Java files is then used to turn the mapping files into generated Java code under the model/target/generated/src directory. These classes are all within the ome.model package. A few hand-written Java classes can also be found in components/model/src/ome/model/internal\(^11\).

---

\(^3\) https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/model
\(^4\) https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/dsl
\(^5\) https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/blitz/resources/templates
\(^6\) https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/src/omero/util/IceMapper.java
\(^7\) http://www.hibernate.org
\(^8\) https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/model/resources/mappings/acquisition.ome.xml
\(^9\) https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/model/resources/mappings
\(^10\) https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/dsl/src
\(^11\) https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/model/src/ome/model/internal
The build-schema ant target takes the generated ome.model classes as input and generates the sql/psql\textsuperscript{12} scripts which get used by omero db script to generate a working OMERO database. Files named like OMEROVERSION__PATCH.sql are hand-written update scripts.

The primary consumer of the ome.model classes at runtime is the components/server\textsuperscript{13} component.

The above classes are considered the internal server code, and are the only objects which can take part in Hibernate transactions.

External to the server code is the “blitz” layer. These classes are in the omero.model package. They are generated by another call to the DSL ant task in order to generate the Java, Python, C++, and Ice files under components/blitz/generated.

The generated Ice files along with the hand-written Ice files from components/blitz/resources/omero\textsuperscript{14} are then run through the slice2cpp, slice2java, and slice2py command-line utilities in order to generate source code in each of these languages. Clients pass in instances of these omero.model (or in the case of C++, omero::model) objects. These are transformed to ome.model objects, and then persisted to the database.

If we take a concrete example, a C++ client might create an Image via new omero::model::ImageI(). The “I” suffix represents an “implementation” in the Ice naming scheme and this subclasses from omero::model::Image. This can be remotely passed to the server which will be deserialized as an omero.model.ImageI object. This will then get converted to an ome.model.core.Image, which can finally be persisted to the database.

**Keywords**

Some words are not allowed as properties/fields of OMERO types. These include:

- id
- version
- details
- … any SQL keyword

### 17.1.2 Improving generated data objects

**Constructors**

Two special constructors are generated for each model object. One is for creating proxy instances, and the other is for filling all NOT-NULL fields:

```java
Pixels p_proxy = new Pixels(Long, boolean);
Pixels p_filled = new Pixels(ome.model.core.Image, ome.model.enums.PixelsType,
   java.lang.Integer, java.lang.Integer, java.lang.Integer, java.lang.Integer, java.lang.Integer,
   java.lang.String, ome.model.enums.DimensionOrder, ome.model.core.PixelsDimensions);
```

The first should almost always be used as: new Pixels(5L, false). Passing in an argument of true would imply that this object is actually loaded, and therefore the server would attempt to null all the fields on your object. See below for a discussion on loadedness.

In the special case of Enumerations, a constructor is generated which takes the value field for the enumeration:

```java
Format file_format = new Format("text/plain");
```

Further, this is the only example of a managed object which will be loaded by the server without its id. This allows applications to record only the string “text/plain” and not need to know the actual id value for “text/plain”.

\textsuperscript{12}https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/sql/psql
\textsuperscript{13}https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/server
\textsuperscript{14}https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/blitz/resources/omero

17.1. OME-Remote Objects 510
Details

Each table in the database has several columns handling low-level matters such as security, ownership, and provenance. To hide some of these details in the object model, each IObject instance contains an ome.model.internal.Details instance.

Details works something like unix’s `stat`:

```
/Types/Images>ls -ltrAG
total 0
-rw------- 1 josh 0 2006-01-25 20:40 Image1
-rw------- 1 josh 0 2006-01-25 20:40 Image2
-rw------- 1 josh 0 2006-01-25 20:40 Image3
-rw-r--r-- 1 josh 0 2006-01-25 20:40 Image100
/Types/Images>stat Image1
File: 'Image1'
   Size: 0 Blocks: 0 IO Block: 4096 regular empty file
Device: 1602h/5634d Inode: 376221 Links: 1
Access: (0600/-rw-------) Uid: ( 1003/ josh) Gid: ( 1001/ ome)
Modify: 2006-01-25 20:40:30.000000000 +0100
Change: 2006-01-25 20:40:30.000000000 +0100

```

though it can also store arbitrary other attributes (meta-metadata, so to speak) about our model instances. See Dynamic methods below for more information.

The main methods on Details are:

```
Permissions Details.getPermissions();
List Details.getUpdates();
Event Details.getCreationEvent();
EventDetails.getUpdatEvent();
Experimenter Details.getOwner();
ExperimenterGroup Details.getGroup();
ExternalInfo getExternalInfo();
```

though some of the methods will return null, if that column is not available for the given object. See Interfaces below for more information.

Consumers of the API are encouraged to pass around Details instances rather than specifying particulars, like:

```
if (securitySystem.allowLoad(Project.class, project.getDetails())) {} 
// and not
if (project.getDetails().getPermissions().isGranted(USER,READ) && project.getDetails().getOwner().getId( myId )) {…}
```

This should hopefully save a good deal of re-coding if we move to true ACL rather than the current filesystem-like access control.

Because it is a field on every type, Details is also on the list of keywords in the type language (above).

Interfaces

To help work with the generated objects, several interfaces are added to their “implements” clause:
<table>
<thead>
<tr>
<th>Property</th>
<th>Applies_to</th>
<th>Interface</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>owner</td>
<td>! global</td>
<td>! global</td>
<td>need sudo</td>
</tr>
<tr>
<td>group</td>
<td>! global</td>
<td></td>
<td>need sudo</td>
</tr>
<tr>
<td>version</td>
<td>! immutable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>creationEvent</td>
<td>! global</td>
<td>! global &amp; ! immutable</td>
<td></td>
</tr>
<tr>
<td>updateEvent</td>
<td>! global</td>
<td></td>
<td></td>
</tr>
<tr>
<td>permissions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>externalInfo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>Named</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td></td>
<td>Described</td>
<td></td>
</tr>
<tr>
<td>linkedAnnotationList</td>
<td></td>
<td>IAnnotated</td>
<td></td>
</tr>
</tbody>
</table>

For example, `ome.model.meta.Experimenter` is a “global” type, therefore it has no `Details.owner` field. In order to create this type of object, you will either need to have admin privileges, or in some cases, use the `ome.api.IAdmin` interface directly (in the case of enums, you will need to use the `ome.api.ITypes` interface).

**Inheritance**

Inheritance is supported in the object model. The superclass relationships can be defined simply in the mapping files. One example is the annotation hierarchy in `components/model/resources/mappings/annotations.ome.xml`. Hibernate supports this polymorphism, and will search all subclasses when a superclass is returned. **However**, due to Hibernate’s use of bytecode-generated proxies, testing for class equality is not always straightforward.

Hibernate uses CGLIB and Javassist and similar bytecode generation to perform much of its magic. For these bytecode generated objects, the `getClass()` method returns something of the form “`ome.model.core.Image_$$.javassist`” which cannot be passed back into Hibernate. Instead, we must first parse that class String with `Utils#trueClass()`.

**Model report objects**

To support the Collection Counts requirement in which users would like to know how many objects are in a collection by owner, it was necessary to add read-only `Map<String, Long>` fields to all objects with links. See the Collection counts page for more information.

**Dynamic methods**

Finally, because not all programming fits into the static programming frame, the object model provides several methods for working dynamically with all IObject subclasses.

**fieldSet / putAt / retrieve**

Each model class contains a public final static String for each field in that class (superclass fields are omitted). A copy of all these fields is available through `fieldSet()`. This field identifier can be used in combination with the `putAt` and `retrieve` methods to store arbitrary data in a class instance. Calls to `putAt/retrieve` with a string found in `fieldSet` delegate to the traditional getters/setters. Otherwise, the value is stored in lazily-initialized `Map` (if no data is stored, the map is `null`).

**acceptFilter**

An automation of calls to `putAt / retrieve` can be achieved by implementing an ome.util.Filter. A Filter is a VisitorPattern-like interface which not only visits every field of an object, but also has the chance to replace the field value with an arbitrary other value. Much of the internal functionality in OMERO is achieved through filters.

---

15 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/model/resources/mappings/annotations.ome.xml
16 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/model/src/ome/util/Utils.java
Limitations

- The filter methods override all standard checks such as IObject#isLoaded and so null-pointer exceptions etc. may be thrown.
- The types stored in the dynamic map currently do not propagate to the OMERO.blitz model objects, since not all java.lang.Objects can be converted.

17.1.3 Entity lifecycle

These additions make certain operations on the model objects easier and cleaner, but they do not save the developer from understanding how each object interacts with Hibernate. Each object has a defined lifecycle and it is important to know both the origin (client, server, or backend) as well as its current state to understand what will and can happen with it.

States

Each instance can be found in one of several states. Quickly, they are:

- **transient** The entity has been created ("new Image()") and not yet shown to the backend.
- **persistent** The entity has been stored in the database and has a non-null id (IObject.getId()). Here Hibernate differentiates between detached, managed, and deleted entities. Detached entities do not take part in lazy-loading or dirty detection like managed entities do. They can, however, be re-attached (made "managed"). Deleted entities cannot take part in most of the ORM activities, and exceptions will be thrown if they are encountered.

- **unloaded (a reference, or proxy)** To solve the common problem of lazy loading exceptions found in many Hibernate applications, we have introduced the concept of unloaded proxy objects which are objects with all fields nulled other than the id. Attempts to get or set any other property will result in an exception. The backend detects these proxies and restores their value before operating on the graph. There are two related states for collections – null which is completely unloaded, and filtered in which certain items have been removed (more on this below).

Identity, references, and versions

Critical for understanding these states is understanding the concepts of identity and versioning as it relates to ORM. Every object has an id field that if created by the backend will not be null. However, every table does not have a primary key field – subclasses contain a foreign key link to their superclass. Therefore all objects without an id are assumed to be non-persistent (i.e. transient).

Though the id cannot be the sole decider of equality since there are issues with the Java definition of equals() and hashCode(), we often perform lookups based on the class and id of an instance. Here again caution must be taken not to unintentionally use a possibly bytecode-generated subclass. See the discussion under Inheritance above.

Class/id-based lookup is in fact so useful that it is possible to take an model object and call obj.unload() to have a “reference” – essentially a placeholder for a model object that contains only an id. Calls to any accessor other than get/setId will throw an exception. An object can be tested for loadedness with obj.isLoaded().

A client can use unloaded instances to inform the backend that a certain information is not available and should be filled in server-side. For example, a user can do the following:

```java
Project p = new Project();
Dataset d = new Dataset( new Long(1), false); // this means create an already unloaded instance
p.linkDataset(d);
iUpdate.saveObject(p);
```

The server, in turn, also uses references to replace backend proxies that would otherwise throw LazyInitializationExceptions on serialization. Clients, therefore, must code with the expectation that the leaves in an object graph may be unloaded. Extending a query with “outer join fetch” will cause these objects to be loaded as well. For example:

```sql
select p from Project p
  left outer join fetch p.datasetLinks as links
  left outer join fetch links.child as dataset
```
but eventually in the complex OME metadata graph, it is certain that something will remain unloaded.

Versions are the last piece to understanding object identity. Two entities with the same id should not be considered equal if they have differing versions. On each write operation, the version of an entity is incremented. This allows us to perform optimistic locking so that two users do not simultaneously edit the same object. That works so:

1. User A and User B retrieve Object X id=1, version=0.
2. User A edits Object X and saves it. Version is incremented to 1.
3. User B edits Object X and tries to save it. The SQL generated is: UPDATE table SET value = newvalue WHERE id = 1 and version = 0; which updates no rows.
4. The fact that no rows were altered is seen by the backend and an OptimisticLockException is thrown.

Identity and versioning make working with the object model difficult sometimes, but guarantee that our data is never corrupted.

Note: There is one exception to this discussed below under Links.
17.1.4 Working with the object model

With these states in mind, it is possible to start looking at how to actually use model objects. From the point of view of the server, everything is either an assertion of an object graph (a “write”) or a request for an object graph (a “read”), whether they are coming from an RMI client, an OMERO.blitz client, or even being generated internally.

Writing

Creating new objects is as simple as instantiating objects and linking them together. If all NOT-NULL fields are not filled, then a ValidationException will be thrown by the server:

```java
IUpdate update = new ServiceFactory().getUpdateService();
Image i = new Image();
try {
    update.saveObject(i);
    catch (ValidationException ve) {
        // not ok.
    }
    i.setName("image");
    return update.saveAndReturnObject(i); // ok.
```

Otherwise, the returned value will be the Image with its id field filled. This works on arbitrarily complex graphs of objects:

```java
Image i = new Image("image-name"); // This constructor exists because "name" is the only required field.
Dataset d = new Dataset("dataset-name");
TagAnnotation tag = new TagAnnotation();
tag.setTextValue("some-tag");
i.linkDataset(d);
i.linkAnnotation(tag);
update.saveAndReturnObject(i);
```

Reading

Reading is a similarly straightforward operation. From a simple id-based lookup, iQuery.get(Experimenter.class, 1L) to a search for an arbitrarily complex graph:

```java
Image i = iQuery.findByQuery("select i from Image i "+
    "join fetch i.datasetLinks as dlinks "+
    "join fetch i.annotationLinks as alinks "+
    "join fetch i.details.owner as owner "+
    "join fetch owner.details.creationEvent "+
    "where i.id = :id", new Parameters().addId(1L));
```

In the return graph, you are guaranteed that any two instances of the same class with the same id are the same object. For example:

```java
Image i = ...; // From query
Dataset d = i.linkedDatasetList().get(0);
Image i2 = d.linkedImageList().get(0);
if (i.getId().equals(i2.getId())) {
    assert i == i2 : "Instances must be referentially equal";
}
```

Reading and writing

Complications arise when you try to mix objects from different read operations because of the difference in equality. In all but the most straightforward applications, references to IObject instances from different return graphs will start to intermingle. For
example, when a user logs in, you might query for all Projects belonging to the user:

```java
List<Project> projects = iQuery.findAllByQuery("select p from Project p where p.details.owner.omeName = someUser", null);
Long id = p.getId();
```

Later you might query for Datasets, and be returned some of the same Projects again within the same graph. You have now possibly got two versions of the Project with a given id within your application. And if one of those Projects has a new Dataset reference, then Hibernate would not know whether the object should be removed or not.

```java
Project oldProject = …; // Acquired from first query
// Do some other work
Dataset dataset = iQuery.findByQuery("select d from Dataset d "+
 "join fetch d.projectsLinks links "+
 "join fetch links.parent "+
 "where d.id = :id", new Parameters().addId(5L));
Project newProject = dataset.linkedProjectList().get(0);
assert newProject.getId().equals(oldProject.getId()) : "same object";
assert newProject.sizeOfDatasetLinks() == oldProject.sizeOfDatasetLinks() :
 "if this is false, then saving oldProject is a problem";
```

Without optimistic locks, trying to save oldProject would cause whatever Datasets were missing from it to be removed from newProject as well. Instead, an `OptimisticLockException` is thrown if a user tries to change an older reference to an entity. Similar problems also arise in multi-user settings, when two users try to access the same object, but it is not purely due to multiple users or even multiple threads, but simply due to stale state.

**Note:** There is an issue with multiple users in which a `SecurityViolation` is thrown instead of an `OptimisticLockException`.

Various techniques to help to manage these duplications are:

- Copy all data to your own model.
- Return unloaded objects wherever possible.
- Be very careful about the operations you commit and about the order they take place in.
- Use a `ClientSession`.

Lazy loading

An issue related to identity is lazy loading. When an object graph is requested, Hibernate loads only the objects which are directly requested. All others are replaced with proxy objects. Within the Hibernate session, these objects are “active” and if accessed, they will be automatically loaded. This is taken care of by the first-level cache, and is also the reason that referential equality is guaranteed within the Hibernate session. Outside of the session however, the proxies can no longer be loaded and so they cannot be serialized to the client.

Instead, as the return value passes through OMERO’s AOP layer, they get disconnected. Single-valued fields are replaced by an unloaded version:

```java
OriginalFile ofile = …; // Object to test
if ( ! Hibernate.isInitialized( ofile.getFormat() ) ) {
   ofile.setFormat( new Format( ofile.getFormat().getId(), false) );
}
```

Multi-valued fields, or collections, are simply nulled. In this case, the `sizeOfXXX` method will return a value less than zero:

```java
Dataset d = …; // Dataset obtained from a query. Didn’t request Projects
assert d.sizeOfProjects() < 0 : "Projects should not be loaded";
```
This is why it is necessary to specify all “join fetch” clauses for instances which are required on the client-side. See [ProxyCleanupFilter](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/tools/hibernate/ProxyCleanupFilter.java) for the implementation.

**Collections**

More than just the nulling during serialization, collections pose several interesting problems. For example, a collection may be filtered on retrieval:

```java
Dataset d = iQuery.findByQuery("select d from Dataset d " +
    "join fetch d.projectLinks links " +
    "where links.parent.id > 2000", null);
```

Some `ProjectDatasetLink` instances have been filtered from the `projectLinks` collection. If the client decides to save this collection back, there is no way to know that it is incomplete, and Hibernate will remove the missing Projects from the Dataset. It is the developer’s responsibility to know what state a collection is in. In the case of links, discussed below, one solution is to use the link objects directly, even if they are largely hidden with the API, but the problem remains for 1-N collections.

**Links**

A special form of links collection model the many-to-many relationship between two other objects. A Project can contain any number of Datasets, and a Dataset can be in any number of Projects. This is achieved by `ProjectDatasetLinks`, which have a Project “parent” and a Dataset “child” (the parent/child terms are somewhat arbitrary but are intended to fit roughly with the users’ expectations for those types).

It is possible to both add and remove a link directly:

```java
ProjectDatasetLink link = new ProjectDatasetLink();
link.setParent( someProject );
link.setChild( someDataset );
link = update.saveAndReturnObject( link );

// someDataset is now included in someProject
update.deleteObject(link);
// or update.deleteObject(new ProjectDatasetLink(link.getId(), false)); // a proxy

// Now the Dataset is not included,
// ___unless___ there was already another link.
```

However, it is also possible to have the links managed for you:

```java
someProject.linkDataset( someDataset ); // This creates the link
update.saveObject( someProject ); // Notices added link, and saves it

someProject.unlinkDataset( someDataset );
update.saveObject( someProject ); // Notices removal, and deletes it
```

The difficulty with this approach is that `unlinkDataset()` will fail if the `someDataset` which you are trying to remove is not referentially equal. That is:

```java
someProject.linkDataset( someDataset );
updatedProject = update.saveAndReturnObject( someProject );

updatedProject.unlinkDataset( someDataset );
update.saveObject( updateProject ); // will do ___nothing___ !
```
does not work since someDataset is not included in updatedProject, but rather updatedDataset with the same id is. Therefore, it would be necessary to do something along the following lines of:

```java
updatedProject = ...; // As before
for (Dataset updatedDataset : updatedProject.linkedDatasetList() ) {
    if (updatedDataset.getId().equals( someDataset.getId() )) {
        updatedProject.unlinkDataset( updatedDataset );
    }
}
```

The unlink method in this case, removes the link from both the Project.datasetLinks collection as well as from the Dataset.projectLinks collection. Hibernate notices that both collections are in agreement, and deletes the ProjectDatasetLink (this is achieved via the “delete-orphan” annotation in Hibernate). If only one side of the collection has had its link removed, an exception will be thrown.

**Synchronization**

Another important point is that the model objects are in no way synchronized. All synchronization must occur within application code.

### 17.1.5 Limitations

We try to minimize differences between the Model as described by the XML specification and its implementation in the OMERO database but some Objects may behave in a more restricted fashion within OMERO. Examples include:

- ROIs and rendering settings can only belong to one Image

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 17.2 Structured annotations

Structured annotations permit the attachment of data and metadata outside the OMERO data model to certain types within the model. The annotations are designed for individualized use by both sites and tools. Annotations can be attached to multiple instances simultaneously to quickly annotated all entities in a view. Each annotation has a “name” which can be interpreted as a “namespace” by tools, which can filter out all unknown namespaces. Further, to prevent users from overwriting or editing important information, annotations are immutable, but editing can be simulated via copy and delete.

#### 17.2.1 Annotated and annotating types

Each type which can be annotated implements `ome.model.IAnnotated`. Currently, these are:

- Project
- Dataset
- Image
- Pixels
- OriginalFile
- PlaneInfo
- Roi
- Channel
- Annotation and all annotation subtypes in order to form hierarchies
- ScreenPlateWell: Screen, ScreenAcquisition, Plate, Well, Reagent
Annotation hierarchy

Though they largely are all String or numeric values, a hierarchy of annotations makes differentiating between just what interpretation should be given to the annotation. This may eventually include validation of the input string and/or file.

Annotation (A*) .......................... A name field and a description
ListAnnotation .......................... Uses AnnotationAnnotation links to form a list of annotations
BasicAnnotation (A*) .......................... Literal or "primitive" values
  BooleanAnnotation .......................... A simple true/false flag
  TimeStampAnnotation .......................... A date/time
  TermAnnotation .......................... A term used in an ontology
NumericAnnotation (A*) .......................... Floating point and integer values
  DoubleAnnotation
  LongAnnotation
TextAnnotation (A*) .......................... A single text field
CommentAnnotation .......................... A user comment
TagAnnotation .......................... Interpreted as a Web 2.0 "tag" on an object, tags on tags form bundles
XmlAnnotation .......................... An xml snippet attached to some object
MapAnnotation .......................... A set of key-value pairs
TypeAnnotation (A*) .......................... Links some entity to another (possibly to be replaced by <any/>)
FileAnnotation .......................... Uses the Format field on OriginalFile to specify type

A* = abstract

See also:
Schema documentation for Structured Annotations\textsuperscript{18}  Section of the auto-generated schema documentation describing the structured annotations

17.2.2 Names and namespaces

Since arbitrary blobs or clobs can be attached to an entity, it is necessary for clients to have some way to differentiate what it can parse. In many cases, the name might be a simple reminder for a user to find the file s/he has annotated. Applications, however, will most likely want to define a namespace, like http://name-of-application-provider.com/name-of-application/file-type/version. Queries can then be produced which search for the proper namespace or match on a part of the name space:

\begin{verbatim}
  iQuery.findAllByQuery("select annotation from FileAnnotation where " +
  "name like 'http://name-of-application-provider.com/name-of-application/%'");
\end{verbatim}

Tags will most likely begin without a namespace. As a tag gets escalated to a common vocabulary, it might make sense to add a possibly site-specific namespace with more well-defined semantics.

17.2.3 Descriptions

Unlike the previous, ImageAnnotation and DatasetAnnotation types, the new structured annotations do not have a description field. The single description field was limited for multi-user scenarios, and can be fully replaced by TextAnnotations attached to another annotation.

\begin{verbatim}
  FileAnnotation fileAnnotation = ...;
  TextAnnotation description = ...;
  fileAnnotation.linkAnnotation(description);
\end{verbatim}
17.2.4 Immutability

The actual content value of an annotation – the text, long, double, file value, etc – is immutable. Links to and from the annotation, however, can be modified.

Currently the namespace field of annotations is mutable. See #878 for discussion.

17.2.5 Examples

Basics

```java
import ome.model.IAnnotated;
import ome.model.annotations.FileAnnotation;
import ome.model.annotations.TagAnnotation;
import ome.model.core.OriginalFile;
import ome.model.display.Roi;

List<Annotation> list = iAnnotated.linkedAnnotationList();
// do something with list
```

Attaching a tag

```java
TagAnnotation tag = new TagAnnotation();
tag.setTextValue("interesting");

Roi roi = ...; // Some region of interest
ILink link = roi.linkAnnotation(tag);

iUpdate.saveObject(link);
```

Attaching a file

```java
// or attach something new
SimpleFile myOriginalFile = new SimpleFile();
myOriginalFile.setName("output.pdf");
// upload PDF

FileAnnotation annotation = new FileAnnotation();
annotation.setName("http://example.com/myClient/analysisOutput");
annotation.setFile(myOriginalFile);

ILink link = iAnnotated.linkAnnotation(annotation)
link = iUpdate.saveAndReturnObject(link);
```

All write changes are intended to occur through the IUpdate interface, whereas searching should be significantly easier through ome.api.Search than IQuery.

See also: Extending OMERO.server

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

19https://trac.openmicroscopy.org/ome/ticket/878
17.3 Glossary of all OMERO Model Objects

17.3.1 Overview

In navigating the model objects used by the OMERO API and in omero hql it is often useful to look up the names of the object properties and the types of their values. This reference document lists every OMERO model object and their more useful properties, with an emphasis on enumerating every direct relationship among the objects.

17.3.2 Reference

**AcquisitionMode**

Used by: *LogicalChannel.mode*

Properties:
- details.externalInfo: *ExternalInfo* (optional)
- value: *string*, see IEnum\(^{20}\)

**Annotation**

Subclasses: *BasicAnnotation, ListAnnotation, MapAnnotation, TextAnnotation, TypeAnnotation*

Used by: *AnnotationAnnotationLink.child, AnnotationAnnotationLink.parent, ChannelAnnotationLink.child, DatasetAnnotationLink.child, DetectorAnnotationLink.child, DicroicAnnotationLink.child, ExperimenterAnnotationLink.child, ExperimenterGroupAnnotationLink.child, FilesetAnnotationLink.child, FilterAnnotationLink.child, ImageAnnotationLink.child, InstrumentAnnotationLink.child, LightPathAnnotationLink.child, LightSourceAnnotationLink.child, NamespaceAnnotationLink.child, NodeAnnotationLink.child, ObjectiveAnnotationLink.child, OriginalFileAnnotationLink.child, PlaneInfoAnnotationLink.child, PlateAnnotationLink.child, PlateAcquisitionAnnotationLink.child, PlateAnnotationLink.child, ProjectAnnotationLink.child, ReagentAnnotationLink.child, RoiAnnotationLink.child, ScreenAnnotationLink.child, SessionAnnotationLink.child, ShapeAnnotationLink.child, WellAnnotationLink.child*

Properties:
- annotationLinks: *AnnotationAnnotationLink* (multiple)
- description: *text* (optional)
- details.creationEvent: *Event*
- details.externalInfo: *ExternalInfo* (optional)
- details.group: *ExperimenterGroup*
- details.owner: *Experimenter*
- details.updateEvent: *Event*
- name: *string* (optional)
- ns: *string* (optional)
- version: *integer* (optional), see IMutable\(^{21}\)

**AnnotationAnnotationLink**

Used by: *Annotation.annotationLinks*

Properties:
- child: *Annotation*, see ILink\(^{22}\)
- details.creationEvent: *Event*
- details.externalInfo: *ExternalInfo* (optional)
- details.group: *ExperimenterGroup*
- details.owner: *Experimenter*\(^{23}\)

---

\(^{20}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IEnum.html
\(^{21}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
\(^{22}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
details.updateEvent: Event
parent: Annotation, see ILink
version: integer (optional), see IMutable

Arc

Properties:

annotationLinks: LightSourceAnnotationLink (multiple) from LightSource
details.creationEvent: Event from LightSource
details.externalInfo: ExternalInfo (optional) from LightSource
details.group: ExperimenterGroup from LightSource
details.owner: Experimenter from LightSource
details.updateEvent: Event from LightSource
instrument: Instrument from LightSource
lotNumber: string (optional) from LightSource
manufacturer: string (optional) from LightSource
model: string (optional) from LightSource
power.unit: enumeration of Power (optional) from LightSource
power.value: double (optional) from LightSource
serialNumber: string (optional) from LightSource
type: ArcType
version: integer (optional) from LightSource

ArcType

Used by: Arc.type

Properties:

details.externalInfo: ExternalInfo (optional)
value: string, see IEnum

BasicAnnotation

Subclasses: BooleanAnnotation, NumericAnnotation, TermAnnotation, TimestampAnnotation

Properties:

annotationLinks: AnnotationAnnotationLink (multiple) from Annotation
description: text (optional) from Annotation
details.creationEvent: Event from Annotation
details.externalInfo: ExternalInfo (optional) from Annotation
details.group: ExperimenterGroup from Annotation
details.owner: Experimenter from Annotation
details.updateEvent: Event from Annotation
name: string (optional) from Annotation
ns: string (optional) from Annotation
version: integer (optional) from Annotation

23http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
25http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Power.html
26http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IEnum.html
Binning

Used by: DetectorSettings.binning

Properties:

- details.externalInfo: ExternalInfo (optional)
- value: string, see IEnum

BooleanAnnotation

Properties:

- annotationLinks: AnnotationAnnotationLink (multiple) from Annotation
- boolValue: boolean (optional)
- description: text (optional) from Annotation
- details.creationEvent: Event from Annotation
- details.externalInfo: ExternalInfo (optional) from Annotation
- details.group: ExperimenterGroup from Annotation
- details.owner: Experimenter from Annotation
- details.updateEvent: Event from Annotation
- name: string (optional) from Annotation
- ns: string (optional) from Annotation
- version: integer (optional) from Annotation

Channel

Used by: ChannelAnnotationLink.parent, LogicalChannel.channels, Pixels.channels

Properties:

- alpha: integer (optional)
- annotationLinks: ChannelAnnotationLink (multiple)
- blue: integer (optional)
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- green: integer (optional)
- logicalChannel: LogicalChannel
- lookupTable: string (optional)
- pixels: Pixels
- red: integer (optional)
- statsInfo: StatsInfo (optional)
- version: integer (optional), see IMutable

ChannelAnnotationLink

Used by: Channel.annotationLinks

Properties:

- child: Annotation, see ILink
- details.creationEvent: Event

---


17.3. Glossary of all OMERO Model Objects
details.externalInfo: $ExternalInfo$ (optional)
details.group: $ExperimenterGroup$
details.owner: $Experimenter$
details.updateEvent: $Event$
parent: $Channel$, see ILink\(^{30}\)
version: integer (optional), see IMutable\(^{31}\)

**ChannelBinding**

**Used by:** RenderingDef.waveRendering

**Properties:**
- active: boolean
- alpha: integer
- blue: integer
- coefficient: double
details.creationEvent: $Event$
details.externalInfo: $ExternalInfo$ (optional)
details.group: $ExperimenterGroup$
details.owner: $Experimenter$
details.updateEvent: $Event$
family: $Family$
green: integer
inputEnd: double
inputStart: double
lookupTable: string (optional)
noiseReduction: boolean
red: integer
renderingDef: RenderingDef
version: integer (optional), see IMutable\(^{32}\)

**ChecksumAlgorithm**

**Used by:** OriginalFile.hasher

**Properties:**
- details.externalInfo: $ExternalInfo$ (optional)
- value: string, see IEnum\(^{33}\)

**CodomainMapContext**

**Subclasses:** ContrastStretchingContext, PlaneSlicingContext, ReverseIntensityContext

**Used by:** RenderingDef.spatialDomainEnhancement

**Properties:**
- details.creationEvent: $Event$
details.externalInfo: $ExternalInfo$ (optional)
details.group: $ExperimenterGroup$
details.owner: $Experimenter$
details.updateEvent: $Event$

---

\(^{30}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\(^{31}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
\(^{32}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
\(^{33}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IEnum.html
renderingDef: RenderingDef
version: integer (optional), see IMutable\textsuperscript{34}

**CommentAnnotation**

**Properties:**
- annotationLinks: AnnotationAnnotationLink (multiple) from Annotation
description: text (optional) from Annotation
details.creationEvent: Event from Annotation
details.externalInfo: ExternalInfo (optional) from Annotation
details.group: ExperimenterGroup from Annotation
details.owner: Experimenter from Annotation
details.updateEvent: Event from Annotation
name: string (optional) from Annotation
ns: string (optional) from Annotation
textValue: text (optional) from TextAnnotation
version: integer (optional) from Annotation

**ContrastMethod**

Used by: LogicalChannel.contrastMethod

**Properties:**
- details.externalInfo: ExternalInfo (optional)
value: string, see IEnum\textsuperscript{35}

**ContrastStretchingContext**

**Properties:**
- details.creationEvent: Event from CodomainMapContext
details.externalInfo: ExternalInfo (optional) from CodomainMapContext
details.group: ExperimenterGroup from CodomainMapContext
details.owner: Experimenter from CodomainMapContext
details.updateEvent: Event from CodomainMapContext
renderingDef: RenderingDef from CodomainMapContext
version: integer (optional) from CodomainMapContext
xend: integer
xstart: integer
yend: integer
ystart: integer

**Correction**

Used by: Objective.correction

**Properties:**
- details.externalInfo: ExternalInfo (optional)
value: string, see IEnum\textsuperscript{36}

\textsuperscript{34}\url{http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html}
\textsuperscript{35}\url{http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IEnum.html}
\textsuperscript{36}\url{http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IEnum.html}
DBPatch

Properties:

- currentPatch: integer
- currentVersion: string
- details.externalInfo: ExternalInfo (optional)
- finished: timestamp (optional)
- message: string (optional)
- previousPatch: integer
- previousVersion: string

Dataset

Used by: DatasetAnnotationLink.parent, DatasetImageLink.parent, ProjectDatasetLink.child

Properties:

- annotationLinks: DatasetAnnotationLink (multiple)
- description: text (optional)
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- imageLinks: DatasetImageLink (multiple)
- name: string
- projectLinks: ProjectDatasetLink (multiple)
- version: integer (optional), see IMutable

DatasetAnnotationLink

Used by: Dataset.annotationLinks

Properties:

- child: Annotation, see ILink
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- parent: Dataset, see ILink
- version: integer (optional), see IMutable

DatasetImageLink

Used by: Dataset.imageLinks, Image.datasetLinks

Properties:

- child: Image, see ILink
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)

---

37http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
38http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
39http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
40http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
41http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
parent: Dataset, see ILink\(^{42}\)
version: integer (optional), see I Mutable\(^{43}\)

**Detector**

Used by: DetectorAnnotationLink.parent, DetectorSettings.detector, Instrument.detector

**Properties:**

- amplificationGain: double (optional)
- annotationLinks: DetectorAnnotationLink (multiple)
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- gain: double (optional)
- instrument: Instrument
- lotNumber: string (optional)
- manufacturer: string (optional)
- model: string (optional)
- offsetValue: double (optional)
- serialNumber: string (optional)
- type: DetectorType
- version: integer (optional), see I Mutable\(^{44}\)
- voltage.unit: enumeration of ElectricPotential\(^{45}\) (optional)
- voltage.value: double (optional)
- zoom: double (optional)

**DetectorAnnotationLink**

Used by: Detector.annotationLinks

**Properties:**

- child: Annotation, see ILink\(^{46}\)
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- parent: Detector, see ILink\(^{47}\)
- version: integer (optional), see I Mutable\(^{48}\)

**DetectorSettings**

Used by: LogicalChannel.detectorSettings

\(^{42}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\(^{43}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
\(^{44}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
\(^{45}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/ElectricPotential.html
\(^{46}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\(^{47}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\(^{48}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html

17.3. Glossary of all OMERO Model Objects 527
Properties:

- binning: *Binning* (optional)
- details.creationEvent: *Event*
- details.externalInfo: *ExternalInfo* (optional)
- details.group: *ExperimenterGroup*
- details.owner: *Experimenter*
- details.updateEvent: *Event*
- detector: *Detector*
- gain: double (optional)
- integration: integer (optional)
- offsetValue: double (optional)
- readOutRate.unit: enumeration of *Frequency*\(^{49}\) (optional)
- readOutRate.value: double (optional)
- version: integer (optional), see *IMutable*\(^{50}\)
- voltage.unit: enumeration of *ElectricPotential*\(^{51}\) (optional)
- voltage.value: double (optional)
- zoom: double (optional)

**DetectorType**

Used by: *Detector.type*

Properties:

- details.externalInfo: *ExternalInfo* (optional)
- value: string, see *IEnum*\(^{52}\)

**Dichroic**

Used by: *DichroicAnnotationLink.parent, FilterSet.dichroic, Instrument.dichroic, LightPath.dichroic*

Properties:

- annotationLinks: *DichroicAnnotationLink* (multiple)
- details.creationEvent: *Event*
- details.externalInfo: *ExternalInfo* (optional)
- details.group: *ExperimenterGroup*
- details.owner: *Experimenter*
- details.updateEvent: *Event*
- instrument: *Instrument*
- lotNumber: string (optional)
- manufacturer: string (optional)
- model: string (optional)
- serialNumber: string (optional)
- version: integer (optional), see *IMutable*\(^{53}\)

**DichroicAnnotationLink**

Used by: *Dichroic.annotationLinks*

Properties:

- \(^{49}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Frequency.html
- \(^{50}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
- \(^{52}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IEnum.html
- \(^{53}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html

17.3. Glossary of all OMERO Model Objects
DimensionOrder

Used by: Pixels.dimensionOrder

Properties:

  details.externalInfo: ExternalInfo (optional)
  value: string, see IEnum

DoubleAnnotation

Properties:

  annotationLinks: AnnotationAnnotationLink (multiple) from Annotation
  description: text (optional) from Annotation
  details.creationEvent: Event from Annotation
  details.externalInfo: ExternalInfo (optional) from Annotation
  details.group: ExperimenterGroup from Annotation
  details.owner: Experimenter from Annotation
  details.updateEvent: Event from Annotation
  doubleValue: double (optional)
  name: string (optional) from Annotation
  ns: string (optional) from Annotation
  version: integer (optional) from Annotation

Ellipse

Properties:

  annotationLinks: ShapeAnnotationLink (multiple) from Shape
  cx: double (optional)
  cy: double (optional)
  details.creationEvent: Event from Shape
  details.externalInfo: ExternalInfo (optional) from Shape
  details.group: ExperimenterGroup from Shape
  details.owner: Experimenter from Shape
  details.updateEvent: Event from Shape
  fillColor: integer (optional) from Shape
  fillRule: string (optional) from Shape
  fontFamily: string (optional) from Shape
  fontSize.unit: enumeration of Length (optional) from Shape
  fontSize.value: double (optional) from Shape

---

http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IEnum.html
http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
fontStretch: string (optional) from Shape
fontStyle: string (optional) from Shape
fontVariant: string (optional) from Shape
fontWeight: string (optional) from Shape
g: string (optional) from Shape
locked: boolean (optional) from Shape
roi: Roi from Shape
rx: double (optional)
ry: double (optional)
strokeColor: integer (optional) from Shape
strokeDashArray: string (optional) from Shape
strokeDashOffset: integer (optional) from Shape
strokeLineCap: string (optional) from Shape
strokeLineWidth: string (optional) from Shape
strokeLineWidthLimit: integer (optional) from Shape
strokeWidth: enumeration of Length (optional) from Shape
strokeWidth: double (optional) from Shape
textValue: text (optional)
theC: integer (optional) from Shape
theT: integer (optional) from Shape
theZ: integer (optional) from Shape
transform: string (optional) from Shape
vectorEffect: string (optional) from Shape
version: integer (optional) from Shape
visibility: boolean (optional) from Shape

Event


59http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
Properties:

- **containingEvent**: `Event` (optional)
- **details.externalInfo**: `ExternalInfo` (optional)
- **experimenter**: `Experimenter`
- **experimenterGroup**: `ExperimenterGroup`
- **logs**: `EventLog` (multiple)
- **session**: `Session`
- **status**: `string` (optional)
- **time**: `timestamp`
- **type**: `EventType`

### EventLog

Used by: `Event.logs`

Properties:

- **action**: `string`
- **details.externalInfo**: `ExternalInfo` (optional)
- **entityId**: `long`
- **entityType**: `string`
- **event**: `Event`

### EventType

Used by: `Event.type`

Properties:
details.externalInfo: ExternalInfo (optional)
value: string, see IEnum

Experiment

Used by: Image.experiment, MicrobeamManipulation.experiment

Properties:
- description: text (optional)
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- microbeamManipulation: MicrobeamManipulation (multiple)
- type: ExperimentType
- version: integer (optional), see IMutable

ExperimentType

Used by: Experiment.type

Properties:
- details.externalInfo: ExternalInfo (optional)
- value: string, see IEnum

Experimenter

Used by: Annotation.details.owner, AnnotationAnnotationLink.details.owner, Channel.details.owner, ChannelAnnotationLink.details.owner, ChannelBinding.details.owner, CodomainMapContext.details.owner, Dataset.details.owner, DatasetAnnotationLink.details.owner, DatasetImageLink.details.owner, Detector.details.owner, DetectorAnnotationLink.details.owner, DetectorSettings.details.owner, Dichroic.details.owner, DichroicAnnotationLink.details.owner, Event.experimenter, Experiment.details.owner, Experimenter/AnnotationLink.details.owner, Experimenter/AnnotationLink.parent, ExperimenterGroupAnnotationLink.details.owner, ExternalInfo.details.owner, Fileset.details.owner, FilesetAnnotationLink.details.owner, FilesetItem.details.owner, FilesetJobLink.details.owner, Filter.details.owner, FilterAnnotationLink.details.owner, FilterSet.details.owner, FilterSetEmissionFilterLink.details.owner, FilterSetExcitationFilterLink.details.owner, GroupExperimenterMap.child, Image.details.owner, ImageAnnotationLink.details.owner, ImagingEnvironment.details.owner, Instrument.details.owner, InstrumentAnnotationLink.details.owner, Job.details.owner, JobOriginalFileLink.details.owner, LightPath.details.owner, LightPathAnnotationLink.details.owner, LightPathEmissionFilterLink.details.owner, LightPathExcitationFilterLink.details.owner, LightSettings.details.owner, LightSource.details.owner, LightSourceAnnotationLink.details.owner, Link.details.owner, LogicalChannel.details.owner, MicrobeamManipulation.details.owner, Microscope.details.owner, NamespaceAnnotationLink.details.owner, NodeAnnotationLink.details.owner, OTF.details.owner, Objective.details.owner, ObjectiveAnnotationLink.details.owner, ObjectiveSettings.details.owner, OriginalFile.details.owner, OriginalFileAnnotationLink.details.owner, Pixels.details.owner, PixelsOriginalFileMap.details.owner, PlaneInfo.details.owner, PlaneInfoAnnotationLink.details.owner, Plate.details.owner, PlateAcquisition.details.owner, PlateAcquisitionAnnotationLink.details.owner, PlateAnnotationLink.details.owner, Project.details.owner, ProjectAnnotationLink.details.owner, ProjectDatasetLink.details.owner, QuantumDef.details.owner, Reagent.details.owner, ReagentAnnotationLink.details.owner, RenderingDef.details.owner, Roi.details.owner, RoiAnnotationLink.details.owner, Screen.details.owner, ScreenAnnotationLink.details.owner, ScreenPlateLink.details.owner, Session.owner, SessionAnnotationLink.details.owner, Shape.details.owner, ShapeAnnotationLink.details.owner, ShareMember.child, StageLabel.details.owner, StatsInfo.details.owner, Thumbail.details.owner, TransmittanceRange.details.owner, Well.details.owner, WellAnnotationLink.details.owner, WellReagentLink.details.owner, WellSample.details.owner

Properties:

60http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IEnum.html
61http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html

17.3. Glossary of all OMERO Model Objects 532
annotationLinks: ExperimenterAnnotationLink (multiple)
details.externalInfo: ExternalInfo (optional)
email: string (optional)
firstName: string
groupExperimenterMap: GroupExperimenterMap (multiple)
institution: string (optional)
lastName: string
ldap: boolean
middleName: string (optional)
omeName: string
version: integer (optional), see IMutable

ExperimenterAnnotationLink

Used by: Experimenter.annotationLinks

Properties:

child: Annotation, see ILink

details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
parent: Experimenter, see ILink

version: integer (optional), see IMutable

ExperimenterGroup

Used by: Annotation.details.group, AnnotationAnnotationLink.details.group, Channel.details.group, ChannelAnnotationLink.details.group, ChannelBinding.details.group, CodomainMapContext.details.group, Dataset.details.group, DatasetAnnotationLink.details.group, Detector.details.group, Detector AnnotationLink.details.group, DetectorSettings.details.group, Dichroic.details.group, DichroicAnnotationLink.details.group, Event.experimenterGroup, Experiment.details.group, ExperimenterAnnotationLink.details.group, ExperimenterGroupAnnotationLink.details.group, ExperimenterGroupAnnotationLink.parent, ExternalInfo.details.group, Fileset.details.group, FilesetAnnotationLink.details.group, FilesetEntry.details.group, FilesetJobLink.details.group, Filter.details.group, FilterAnnotationLink.details.group, FilterSet.details.group, FilterSetEmissionFilterLink.details.group, FilterSetExcitationFilterLink.details.group, GroupExperimenterMap.parent, Image.details.group, ImageAnnotationLink.details.group, ImagingEnvironment.details.group, Instrument.details.group, InstrumentAnnotationLink.details.group, Job.details.group, JobOriginalFile.details.group, LightPath.details.group, LightPathAnnotationLink.details.group, LightPathEmissionFilterLink.details.group, LightPathExcitationFilterLink.details.group, LightSettings.details.group, LightSource.details.group, LightSourceAnnotationLink.details.group, Link.details.group, LogicalChannel.details.group, MicrobeamManipulation.details.group, Microscope.details.group, NamespaceAnnotationLink.details.group, NodeAnnotationLink.details.group, OTF.details.group, Objective.details.group, ObjectiveAnnotationLink.details.group, ObjectiveSettings.details.group, OriginalFile.details.group, OriginalFileAnnotationLink.details.group, Pixels.details.group, PixelsOriginalFileMap.details.group, Plane.details.group, PlaneAnnotationLink.details.group, Plate Acquisition.details.group, Plate AcquisitionAnnotationLink.details.group, PlateDetailsGroup.annotationLink, Project.details.group, ProjectAnnotationLink.details.group, ProjectDatasetLink.details.group, QuantumDef.details.group, Reagent.details.group, ReagentAnnotationLink.details.group, RenderingDef.details.group, ROIGroup.annotationLink, ROIAnnotationLink.details.group, Screen.details.group, ScreenAnnotationLink.details.group, ScreenPlateLink.details.group, SessionAnnotationLink.details.group, Shape.details.group, ShapeAnnotationLink.details.group, ShareGroup.annotationLink, StageLabel.details.group, StatsInfo.details.group, Thumbnail.details.group, TransmittanceRange.details.group, Well.details.group, WellAnnotationLink.details.group, WellReagentLink.details.group, WellSample.details.group

Properties:

63http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
64http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
65http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html

17.3. Glossary of all OMERO Model Objects 533
annotationLinks: ExperimenterGroupAnnotationLink (multiple)
config: list (multiple)
description: text (optional)
details.externalInfo: ExternalInfo (optional)
groupExperimenterMap: GroupExperimenterMap (multiple)
ldap: boolean
name: string
version: integer (optional), see IMutable67

ExperimenterGroupAnnotationLink

Used by: ExperimenterGroup.annotationLinks

Properties:

child: Annotation, see ILink68
details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
parent: ExperimenterGroup, see ILink69
version: integer (optional), see IMutable70

ExternalInfo


68http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
69http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
70http://downloads.openmicroscopy.org/latest/omero/api/ome/model/I Mutable.html

17.3. Glossary of all OMERO Model Objects 534
Properties:

details.creationEvent: Event

details.externalInfo: ExternalInfo (optional)

details.group: ExperimenterGroup

details.owner: Experimenter

entityId: long

entityType: string

lsid: string (optional)

uuid: string (optional)

Family

Used by: ChannelBinding.family

Properties:

details.externalInfo: ExternalInfo (optional)

value: string, see IEnum71

Filament

Properties:

annotationLinks: LightSourceAnnotationLink (multiple) from LightSource

details.creationEvent: Event from LightSource

details.externalInfo: ExternalInfo (optional) from LightSource

details.group: ExperimenterGroup from LightSource

details.owner: Experimenter from LightSource

details.updateEvent: Event from LightSource

instrument: Instrument from LightSource

lotNumber: string (optional) from LightSource

manufacturer: string (optional) from LightSource

model: string (optional) from LightSource

power.unit: enumeration of Power72 (optional) from LightSource

power.value: double (optional) from LightSource

serialNumber: string (optional) from LightSource

type: FilamentType

version: integer (optional) from LightSource

FilamentType

Used by: Filament.type

71http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IEnum.html

72http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Power.html
Properties:
  - details.externalInfo: *ExternalInfo* (optional)
  - value: *string*, see IEnum

**FileAnnotation**

Properties:
  - annotationLinks: *AnnotationAnnotationLink* (multiple) from *Annotation*
  - description: *text* (optional) from *Annotation*
  - details.creationEvent: *Event* from *Annotation*
  - details.externalInfo: *ExternalInfo* (optional) from *Annotation*
  - details.group: *ExperimenterGroup* from *Annotation*
  - details.owner: *Experimenter* from *Annotation*
  - details.updateEvent: *Event* from *Annotation*
  - file: *OriginalFile* (optional)
  - name: *string* (optional) from *Annotation*
  - ns: *string* (optional) from *Annotation*
  - version: *integer* (optional) from *Annotation*

**Fileset**

Used by: FilesetAnnotationLink.parent, FilesetEntry.fileset, FilesetJobLink.parent, Image.fileset

Properties:
  - annotationLinks: *FilesetAnnotationLink* (multiple)
  - details.creationEvent: *Event*
  - details.externalInfo: *ExternalInfo* (optional)
  - details.group: *ExperimenterGroup*
  - details.owner: *Experimenter*
  - details.updateEvent: *Event*
  - images: *Image* (multiple)
  - jobLinks: *FilesetJobLink* (multiple)
  - templatePrefix: *text*
  - usedFiles: *FilesetEntry* (multiple)
  - version: *integer* (optional), see IMutable

**FilesetAnnotationLink**

Used by: Fileset.annotationLinks

Properties:
  - child: *Annotation*, see ILink
  - details.creationEvent: *Event*
  - details.externalInfo: *ExternalInfo* (optional)
  - details.group: *ExperimenterGroup*
  - details.owner: *Experimenter*
  - details.updateEvent: *Event*
  - parent: *Fileset*, see ILink
  - version: *integer* (optional), see IMutable

---

73 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IEnum.html
74 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
75 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
76 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
77 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
FilesetEntry

Used by: Fileset.usedFiles, OriginalFile.filesetEntries

Properties:

- clientPath: text
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- fileset: Fileset
- originalFile: OriginalFile
- version: integer (optional), see IMutable\textsuperscript{78}

FilesetJobLink

Used by: Fileset.jobLinks

Properties:

- child: Job, see ILink\textsuperscript{79}
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- parent: Fileset, see ILink\textsuperscript{80}
- version: integer (optional), see IMutable\textsuperscript{81}

Filter

Used by: FilterAnnotationLink.parent, FilterSetEmissionFilterLink.child, FilterSetExcitationFilterLink.child, Instrument.filter, LightPathEmissionFilterLink.child, LightPathExcitationFilterLink.child

Properties:

- annotationLinks: FilterAnnotationLink (multiple)
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- emissionFilterLink: FilterSetEmissionFilterLink (multiple)
- excitationFilterLink: FilterSetExcitationFilterLink (multiple)
- filterWheel: string (optional)
- instrument: Instrument
- lotNumber: string (optional)
- manufacturer: string (optional)
- model: string (optional)
- serialNumber: string (optional)
- transmittanceRange: TransmittanceRange (optional)
- type: FilterType (optional)

\textsuperscript{78}http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
\textsuperscript{79}http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\textsuperscript{80}http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\textsuperscript{81}http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html

17.3. Glossary of all OMERO Model Objects 537
version: integer (optional), see IMutable

**FilterAnnotationLink**

Used by: *Filter.annotationLinks*

**Properties:**
- child: *Annotation*, see ILink
- details.creationEvent: *Event*
- details.externalInfo: *ExternalInfo* (optional)
- details.group: *ExperimenterGroup*
- details.owner: *Experimenter*
- details.updateEvent: *Event*
- parent: *Filter*, see ILink
- version: integer (optional), see IMutable

**FilterSet**

Used by: *FilterSetEmissionFilterLink.parent*, *FilterSetExcitationFilterLink.parent*, *Instrument.filterSet*, *LogicalChannel.filterSet*, *OTF.filterSet*

**Properties:**
- details.creationEvent: *Event*
- details.externalInfo: *ExternalInfo* (optional)
- details.group: *ExperimenterGroup*
- details.owner: *Experimenter*
- details.updateEvent: *Event*
- dichroic: *Dichroic* (optional)
- emissionFilterLink: *FilterSetEmissionFilterLink* (multiple)
- excitationFilterLink: *FilterSetExcitationFilterLink* (multiple)
- instrument: *Instrument*
- lotNumber: string (optional)
- manufacturer: string (optional)
- model: string (optional)
- serialNumber: string (optional)
- version: integer (optional), see IMutable

**FilterSetEmissionFilterLink**

Used by: *Filter.emissionFilterLink*, *FilterSet.emissionFilterLink*

**Properties:**
- child: *Filter*, see ILink
- details.creationEvent: *Event*
- details.externalInfo: *ExternalInfo* (optional)
- details.group: *ExperimenterGroup*
- details.owner: *Experimenter*
- details.updateEvent: *Event*
- parent: *FilterSet*, see ILink

---

84 [http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html](http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html)
87 [http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html](http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html)

17.3. Glossary of all OMERO Model Objects 538
version: integer (optional), see IMutable

**FilterSetExcitationFilterLink**

Used by: `Filter.excitiationFilterLink`, `FilterSet.excitationFilterLink`

**Properties:**
- child: `Filter`, see ILink
- details.creationEvent: `Event`
- details.externalInfo: `ExternalInfo` (optional)
- details.group: `ExperimenterGroup`
- details.owner: `Experimenter`
- details.updateEvent: `Event`
- parent: `FilterSet`, see ILink
- version: integer (optional), see IMutable

**FilterType**

Used by: `Filter.type`

**Properties:**
- details.externalInfo: `ExternalInfo` (optional)
- value: string, see IEnum

**Format**

Used by: `Image.format`

**Properties:**
- details.externalInfo: `ExternalInfo` (optional)
- value: string, see IEnum

**GenericExcitationSource**

**Properties:**
- annotationLinks: `LightSourceAnnotationLink` (multiple) from `LightSource`
- details.creationEvent: `Event` from `LightSource`
- details.externalInfo: `ExternalInfo` (optional) from `LightSource`
- details.group: `ExperimenterGroup` from `LightSource`
- details.owner: `Experimenter` from `LightSource`
- details.updateEvent: `Event` from `LightSource`
- instrument: `Instrument` from `LightSource`
- lotNumber: string (optional) from `LightSource`
- manufacturer: string (optional) from `LightSource`
- map: list (multiple)
- model: string (optional) from `LightSource`
- power.unit: enumeration of `Power` (optional) from `LightSource`
- power.value: double (optional) from `LightSource`

---

89 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
90 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
91 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
92 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
93 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IEnum.html
95 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Power.html

17.3. Glossary of all OMERO Model Objects 539
serialNumber: string (optional) from LightSource
version: integer (optional) from LightSource

GroupExperimenterMap

Used by: Experimenter.groupExperimenterMap, ExperimenterGroup.groupExperimenterMap

Properties:
  child: Experimenter, see ILink96
details.externalInfo: ExternalInfo (optional)
owner: boolean
parent: ExperimenterGroup, see ILINK97
version: integer (optional), see IMutable98

Illumination

Used by: LogicalChannel.illumination

Properties:
  details.externalInfo: ExternalInfo (optional)
  value: string, see IEnum99

Image

Used by: DatasetImageLink.child, Fileset.images, ImageAnnotationLink.parent, Pixels.image, Roi.image, WellSample.image

Properties:
  acquisitionDate: timestamp (optional)
  annotationLinks: ImageAnnotationLink (multiple)
  archived: boolean (optional)
datasetLinks: DatasetImageLink (multiple)
description: text (optional)
details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
experiment: Experiment (optional)
fileset: Fileset (optional)
format: Format (optional)
ingoingEnvironment: ImagingEnvironment (optional)
instrument: Instrument (optional)
name: string
objectiveSettings: ObjectiveSettings (optional)
partial: boolean (optional)
pixels: Pixels (multiple)
rois: Roi (multiple)
series: integer (optional)
stageLabel: StageLabel (optional)
version: integer (optional), see IMutable100

96http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
97http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILINK.html
100http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
wellSamples: WellSample (multiple)

**ImageAnnotationLink**

Used by: Image.annotationLinks

Properties:

- child: Annotation, see ILink
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- parent: Image, see ILink
- version: integer (optional), see IMutable

**ImagingEnvironment**

Used by: Image.imagingEnvironment

Properties:

- airPressure.unit: enumeration of Pressure (optional)
- airPressure.value: double (optional)
- co2percent: double (optional)
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- humidity: double (optional)
- map: list (multiple)
- temperature.unit: enumeration of Temperature (optional)
- temperature.value: double (optional)
- version: integer (optional), see IMutable

**Immersion**

Used by: Objective.immersion

Properties:

- details.externalInfo: ExternalInfo (optional)
- value: string, see IEnum

**ImportJob**

Properties:

- details.creationEvent: Event from Job
- details.externalInfo: ExternalInfo (optional) from Job

---

http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Pressure.html
http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Temperature.html
http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IEnum.html

17.3. Glossary of all OMERO Model Objects
IndexingJob

Properties:

- details.creationEvent: Event from Job
- details.externalInfo: ExternalInfo (optional) from Job
- details.group: ExperimenterGroup from Job
- details.owner: Experimenter from Job
- details.updateEvent: Event from Job
- finished: timestamp (optional) from Job
- groupname: string from Job
- message: string from Job
- originalFileLinks: JobOriginalFileLink (multiple) from Job
- scheduledFor: timestamp from Job
- started: timestamp (optional) from Job
- status: JobStatus from Job
- submitted: timestamp from Job
- type: string from Job
- username: string from Job
- version: integer (optional) from Job

Instrument

Used by: Detector.instrument, Dichroic.instrument, Filter.instrument, FilterSet.instrument, Image.instrument, InstrumentAnnotationLink.parent, LightSource.instrument, OTF.instrument, Objective.instrument

Properties:

- annotationLinks: InstrumentAnnotationLink (multiple)
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- detector: Detector (multiple)
- dichroic: Dichroic (multiple)
- filter: Filter (multiple)
- filterSet: FilterSet (multiple)
lightSource: \textit{LightSource} (multiple)
microscope: \textit{Microscope} (optional)
objective: \textit{Objective} (multiple)
of: \textit{OTF} (multiple)
version: integer (optional), see IMutable\(^{108}\)

\textbf{InstrumentAnnotationLink}

Used by: \textit{Instrument.annotationLinks}

Properties:
\begin{itemize}
  \item child: \textit{Annotation}, see ILink\(^{109}\)
  \item details.creationEvent: \textit{Event}
  \item details.externalInfo: \textit{ExternalInfo} (optional)
  \item details.group: \textit{ExperimenterGroup}
  \item details.owner: \textit{Experimenter}
  \item details.updateEvent: \textit{Event}
  \item parent: \textit{Instrument}, see ILink\(^{110}\)
  \item version: integer (optional), see IMutable\(^{111}\)
\end{itemize}

\textbf{IntegrityCheckJob}

Properties:
\begin{itemize}
  \item details.creationEvent: \textit{Event} from \textit{Job}
  \item details.externalInfo: \textit{ExternalInfo} (optional) from \textit{Job}
  \item details.group: \textit{ExperimenterGroup} from \textit{Job}
  \item details.owner: \textit{Experimenter} from \textit{Job}
  \item details.updateEvent: \textit{Event} from \textit{Job}
  \item finished: \textit{timestamp} (optional) from \textit{Job}
  \item groupname: \textit{string} from \textit{Job}
  \item message: \textit{string} from \textit{Job}
  \item originalFileLinks: \textit{JobOriginalFileLink} (multiple) from \textit{Job}
  \item scheduledFor: \textit{timestamp} from \textit{Job}
  \item started: \textit{timestamp} (optional) from \textit{Job}
  \item status: \textit{JobStatus} from \textit{Job}
  \item submitted: \textit{timestamp} from \textit{Job}
  \item type: \textit{string} from \textit{Job}
  \item username: \textit{string} from \textit{Job}
  \item version: integer (optional) from \textit{Job}
\end{itemize}

\textbf{Job}


Used by: \textit{FilesetJobLink.child}, \textit{JobOriginalFileLink.parent}

Properties:
\begin{itemize}
  \item details.creationEvent: \textit{Event}
  \item details.externalInfo: \textit{ExternalInfo} (optional)
\end{itemize}

\(^{108}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
\(^{109}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\(^{110}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\(^{111}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html

17.3. Glossary of all OMERO Model Objects 543
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
finished: timestamp (optional)
groupname: string
message: string
originalFileLinks: JobOriginalFileLink (multiple)
scheduledFor: timestamp
started: timestamp (optional)
status: JobStatus
submitted: timestamp
type: string
username: string
version: integer (optional), see IMutable

JobOriginalFileLink

Used by: Job.originalFileLinks

Properties:

child: OriginalFile, see ILink

details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
parent: Job, see ILink

version: integer (optional), see IMutable

JobStatus

Used by: Job.status

Properties:


details.externalInfo: ExternalInfo (optional)
value: string, see IEnum

Label

Properties:

anchor: string (optional)
annotationLinks: ShapeAnnotationLink (multiple) from Shape
baselineShift: string (optional)
decoration: string (optional)
details.creationEvent: Event from Shape
details.externalInfo: ExternalInfo (optional) from Shape
details.group: ExperimenterGroup from Shape
details.owner: Experimenter from Shape
details.updateEvent: Event from Shape

112 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
113 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
114 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
direction: string (optional)
fillColor: integer (optional) from Shape
fillRule: string (optional) from Shape
fontFamily: string (optional) from Shape
fontSize.unit: enumeration of Length\(^{117}\) (optional) from Shape
fontSize.value: double (optional) from Shape
fontStretch: string (optional) from Shape
fontStyle: string (optional) from Shape
fontVariant: string (optional) from Shape
fontWeight: string (optional) from Shape
g: string (optional) from Shape
glyphOrientationVertical: integer (optional)
locked: boolean (optional) from Shape
roi: Roi from Shape
strokeColor: integer (optional) from Shape
strokeDashArray: string (optional) from Shape
strokeDashOffset: integer (optional) from Shape
strokeLineCap: string (optional) from Shape
strokeLineJoin: string (optional) from Shape
strokeMiterLimit: integer (optional) from Shape
strokeWidth.unit: enumeration of Length\(^{118}\) (optional) from Shape
strokeWidth.value: double (optional) from Shape
textValue: text (optional)
theC: integer (optional) from Shape
theT: integer (optional) from Shape
theZ: integer (optional) from Shape
transform: string (optional) from Shape
vectorEffect: string (optional) from Shape
version: integer (optional) from Shape
visibility: boolean (optional) from Shape
writingMode: string (optional)
x: double (optional)
y: double (optional)

**Laser**

Properties:

- annotationLinks: `LightSourceAnnotationLink` (multiple) from `LightSource`
- details.creationEvent: `Event` from `LightSource`
- details.externalInfo: `ExternalInfo` (optional) from `LightSource`
- details.group: `ExperimenterGroup` from `LightSource`
- details.owner: `Experimenter` from `LightSource`
- details.updateEvent: `Event` from `LightSource`
- frequencyMultiplication: integer (optional)
- instrument: `Instrument` from `LightSource`
- laserMedium: `LaserMedium`
- lotNumber: string (optional) from `LightSource`
- manufacturer: string (optional) from `LightSource`
- model: string (optional) from `LightSource`
- pockelCell: boolean (optional)

\(^{117}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html

\(^{118}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
LaserMedium

Used by: Laser.laserMedium

Properties:

- details.externalInfo: ExternalInfo (optional)
  - value: string, see IEnum

LaserType

Used by: Laser.type

Properties:

- details.externalInfo: ExternalInfo (optional)
  - value: string, see IEnum

Light Emitting Diode

Properties:

- annotationLinks: LightSourceAnnotationLink (multiple) from LightSource
- details.creationEvent: Event from LightSource
- details.externalInfo: ExternalInfo (optional) from LightSource
- details.group: ExperimenterGroup from LightSource
- details.owner: Experimenter from LightSource
- details.updateEvent: Event from LightSource
- instrument: Instrument from LightSource
- lotNumber: string (optional) from LightSource
- manufacturer: string (optional) from LightSource
- model: string (optional) from LightSource
- power.unit: enumeration of Power (optional) from LightSource
- power.value: double (optional) from LightSource
- serialNumber: string (optional) from LightSource
- version: integer (optional) from LightSource

---

121: http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
124: http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Power.html
LightPath

Used by: LightPathAnnotationLink.parent, LightPathEmissionFilterLink.parent, LightPathExcitationFilterLink.parent, LogicalChannel.lightPath

Properties:

- annotationLinks: LightPathAnnotationLink (multiple)
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- dichroic: Dichroic (optional)
- emissionFilterLink: LightPathEmissionFilterLink (multiple)
- excitationFilterLink: LightPathExcitationFilterLink (multiple)
- version: integer (optional), see IMutable

LightPathAnnotationLink

Used by: LightPath.annotationLinks

Properties:

- child: Annotation, see ILink
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- parent: LightPath, see ILink
- version: integer (optional), see IMutable

LightPathEmissionFilterLink

Used by: LightPath.emissionFilterLink

Properties:

- child: Filter, see ILink
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- parent: LightPath, see ILink
- version: integer (optional), see IMutable

LightPathExcitationFilterLink

Used by: LightPath.excitationFilterLink
Properties:

child: *Filter*, see ILink\(^{132}\)
details.creationEvent: *Event*
details.externalInfo: *ExternalInfo* (optional)
details.group: *ExperimenterGroup*
details.owner: *Experimenter*
details.updateEvent: *Event*
parent: *LightPath*, see ILink\(^{133}\)
version: integer (optional), see IMutable\(^{134}\)

**LightSettings**

Used by: *LogicalChannel.lightSourceSettings*, *MicrobeamManipulation.lightSourceSettings*

Properties:

attenuation: double (optional)
details.creationEvent: *Event*
details.externalInfo: *ExternalInfo* (optional)
details.group: *ExperimenterGroup*
details.owner: *Experimenter*
details.updateEvent: *Event*
lightSource: *LightSource*
microbeamManipulation: *MicrobeamManipulation* (optional)
version: integer (optional), see IMutable\(^{135}\)
wavelength.unit: enumeration of *Length*\(^{136}\) (optional)
wavelength.value: double (optional)

**LightSource**

Subclasses: *Arc*, *Filament*, *GenericExcitationSource*, *Laser*, *LightEmittingDiode*


Properties:

annotationLinks: *LightSourceAnnotationLink* (multiple)
details.creationEvent: *Event*
details.externalInfo: *ExternalInfo* (optional)
details.group: *ExperimenterGroup*
details.owner: *Experimenter*
details.updateEvent: *Event*
instrument: *Instrument*
lotNumber: string (optional)
manufacturer: string (optional)
model: string (optional)
power.unit: enumeration of *Power*\(^{137}\) (optional)
power.value: double (optional)
serialNumber: string (optional)
version: integer (optional), see IMutable\(^{138}\)

\(^{132}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\(^{133}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\(^{134}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
\(^{135}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
\(^{136}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
\(^{137}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Power.html
\(^{138}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
LightSourceAnnotationLink

Used by: LightSource.annotationLinks

Properties:

child: Annotation, see ILink

details.creationEvent: Event

details.externalInfo: ExternalInfo (optional)

details.group: ExperimenterGroup

details.owner: Experimenter

details.updateEvent: Event

parent: LightSource, see ILink

version: integer (optional), see IMutable

Line

Properties:

annotationLinks: ShapeAnnotationLink (multiple) from Shape

details.creationEvent: Event from Shape

details.externalInfo: ExternalInfo (optional) from Shape

details.group: ExperimenterGroup from Shape

details.owner: Experimenter from Shape

details.updateEvent: Event from Shape

fillColor: integer (optional) from Shape

fillRule: string (optional) from Shape

fontFamily: string (optional) from Shape

fontSize.unit: enumeration of Length (optional) from Shape

fontSize.value: double (optional) from Shape

fontStretch: string (optional) from Shape

fontStyle: string (optional) from Shape

fontVariant: string (optional) from Shape

fontWeight: string (optional) from Shape

g: string (optional) from Shape

locked: boolean (optional) from Shape

roi: Roi from Shape

strokeColor: integer (optional) from Shape

strokeDashArray: string (optional) from Shape

strokeDashOffset: integer (optional) from Shape

strokeLineCap: string (optional) from Shape

strokeLineJoin: string (optional) from Shape

strokeMiterLimit: integer (optional) from Shape

strokeWidth.unit: enumeration of Length (optional) from Shape

strokeWidth.value: double (optional) from Shape

textValue: text (optional)

theC: integer (optional) from Shape

theT: integer (optional) from Shape

theZ: integer (optional) from Shape

transform: string (optional) from Shape

vectorEffect: string (optional) from Shape

---

139 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
140 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
141 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
142 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
143 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
version: integer (optional) from Shape
visibility: boolean (optional) from Shape
x1: double (optional)
x2: double (optional)
y1: double (optional)
y2: double (optional)

Link

Properties:
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- version: integer (optional), see IMutable

ListAnnotation

Properties:
- annotationLinks: AnnotationAnnotationLink (multiple) from Annotation
description: text (optional) from Annotation
details.creationEvent: Event from Annotation
details.externalInfo: ExternalInfo (optional) from Annotation
details.group: ExperimenterGroup from Annotation
details.owner: Experimenter from Annotation
details.updateEvent: Event from Annotation
name: string (optional) from Annotation
ns: string (optional) from Annotation
version: integer (optional) from Annotation

LogicalChannel

Used by: Channel.logicalChannel

Properties:
- channels: Channel (multiple)
- contrastMethod: ContrastMethod (optional)
details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
detectorSettings: DetectorSettings (optional)
emissionWave.unit: enumeration of Length
emissionWave.value: double (optional)
excitationWave.unit: enumeration of Length
excitationWave.value: double (optional)
filterSet: FilterSet (optional)
fluor: string (optional)

144 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
145 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
146 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html

17.3. Glossary of all OMERO Model Objects
illuminaton: *Illumination* (optional)
lightPath: *LightPath* (optional)
lightSourceSettings: *LightSettings* (optional)
mode: *AcquisitionMode* (optional)
name: string (optional)
ndFilter: double (optional)
of: *OTF* (optional)
photometricInterpretation: *PhotometricInterpretation* (optional)
pinHoleSize.unit: enumeration of *Length*\(^{147}\) (optional)
pinHoleSize.value: double (optional)
pocketCellSetting: integer (optional)
samplesPerPixel: integer (optional)
version: integer (optional), see *IMutable*\(^{148}\)

**LongAnnotation**

Properties:

- annotationLinks: *AnnotationAnnotationLink* (multiple) from *Annotation*
- description: text (optional) from *Annotation*
- details.creationEvent: *Event* from *Annotation*
- details.externalInfo: *ExternalInfo* (optional) from *Annotation*
- details.group: *ExperimenterGroup* from *Annotation*
- details.owner: *Experimenter* from *Annotation*
- details.updateEvent: *Event* from *Annotation*
- longValue: long (optional)
- name: string (optional) from *Annotation*
- ns: string (optional) from *Annotation*
- version: integer (optional) from *Annotation*

**MapAnnotation**

Properties:

- annotationLinks: *AnnotationAnnotationLink* (multiple) from *Annotation*
- description: text (optional) from *Annotation*
- details.creationEvent: *Event* from *Annotation*
- details.externalInfo: *ExternalInfo* (optional) from *Annotation*
- details.group: *ExperimenterGroup* from *Annotation*
- details.owner: *Experimenter* from *Annotation*
- details.updateEvent: *Event* from *Annotation*
- mapValue: list (multiple)
- name: string (optional) from *Annotation*
- ns: string (optional) from *Annotation*
- version: integer (optional) from *Annotation*

**Mask**

Properties:

- annotationLinks: *ShapeAnnotationLink* (multiple) from *Shape*
- bytes: binary (optional)
- details.creationEvent: *Event* from *Shape*  

\(^{147}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html  
\(^{148}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
Medium

Used by: ObjectiveSettings.medium

Properties:

- details.externalInfo: ExternalInfo (optional)
- value: string, see IEnum

MetadataImportJob

Properties:

- details.creationEvent: Event from Job

---

\[149\] http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html

\[150\] http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html

\[151\] http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IEnum.html
details.externalInfo: ExternalInfo (optional) from Job
details.group: ExperimenterGroup from Job
details.owner: Experimenter from Job
details.updateEvent: Event from Job
finished: timestamp (optional) from Job
groupname: string from Job
message: string from Job
originalFileLinks: JobOriginalFileLink (multiple) from Job
scheduledFor: timestamp from Job
started: timestamp (optional) from Job
status: JobStatus from Job
submitted: timestamp from Job
type: string from Job
username: string from Job
version: integer (optional) from Job
versionInfo: list (multiple)

MicrobeamManipulation

Used by: Experiment.microbeamManipulation, LightSettings.microbeamManipulation

Properties:

description: text (optional)
details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
eperiment: Experiment
lightSourceSettings: LightSettings (multiple)
type: MicrobeamManipulationType
version: integer (optional), see IMutable,153

MicrobeamManipulationType

Used by: MicrobeamManipulation.type

Properties:

details.externalInfo: ExternalInfo (optional)
value: string, see IEnum,153

Microscope

Used by: Instrument.microscope

Properties:

details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
lotNumber: string (optional)

152 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html

17.3. Glossary of all OMERO Model Objects
manufacturer: string (optional)
model: string (optional)
serialNumber: string (optional)
type: MicroscopeType
version: integer (optional), see IMutable

**MicroscopeType**

Used by: Microscope.type

Properties:

- details.externalInfo: ExternalInfo (optional)
- value: string, see IEnum

**Namespace**

Used by: NamespaceAnnotationLink.parent

Properties:

- annotationLinks: NamespaceAnnotationLink (multiple)
- description: text (optional)
- details.externalInfo: ExternalInfo (optional)
- display: boolean (optional)
- displayName: string (optional)
- keywords: list (optional)
- multivalued: boolean (optional)
- name: string
- version: integer (optional), see IMutable

**NamespaceAnnotationLink**

Used by: Namespace.annotationLinks

Properties:

- child: Annotation, see ILink
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- parent: Namespace, see ILink
- version: integer (optional), see IMutable

**Node**

Used by: NodeAnnotationLink.parent, Session.node

Properties:

- annotationLinks: NodeAnnotationLink (multiple)
conn: text
details.externalInfo: ExternalInfo (optional)
down: timestamp (optional)
scale: integer (optional)
sessions: Session (multiple)
up: timestamp
uuid: string
version: integer (optional), see IMutable

NodeAnnotationLink

Used by: Node.annotationLinks

Properties:
  child: Annotation, see ILink
  details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
  parent: Node, see ILink
  version: integer (optional), see IMutable

NumericAnnotation

Subclasses: DoubleAnnotation, LongAnnotation

Properties:
  annotationLinks: AnnotationAnnotationLink (multiple) from Annotation
description: text (optional) from Annotation
details.creationEvent: Event from Annotation
details.externalInfo: ExternalInfo (optional) from Annotation
details.group: ExperimenterGroup from Annotation
details.owner: Experimenter from Annotation
details.updateEvent: Event from Annotation
  name: string (optional) from Annotation
  ns: string (optional) from Annotation
  version: integer (optional) from Annotation

OTF

Used by: Instrument.otf, LogicalChannel.otf

Properties:
  details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
  filterSet: FilterSet (optional)

161 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
162 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
instrument: *Instrument*
objective: *Objective*
opticalAxisAveraged: boolean
path: string
pixelsType: *PixelsType*
sizeX: integer
sizeY: integer
version: integer (optional), see IMutable\(^{164}\)

**Objective**

Used by: *Instrument.objective*, *OTF.objective*, *ObjectiveAnnotationLink.parent*, *ObjectiveSettings.objective*

**Properties:**

- annotationLinks: *ObjectiveAnnotationLink* (multiple)
- calibratedMagnification: double (optional)
- correction: *Correction*
- details.creationEvent: *Event*
- details.externalInfo: *ExternalInfo* (optional)
- details.group: *ExperimenterGroup*
- details.owner: *Experimenter*
- details.updateEvent: *Event*
- immersion: *Immersion*
- instrument: *Instrument*
- iris: boolean (optional)
- lensNA: double (optional)
- lotNumber: string (optional)
- manufacturer: string (optional)
- model: string (optional)
- nominalMagnification: double (optional)
- serialNumber: string (optional)
- version: integer (optional), see IMutable\(^{165}\)
- workingDistance.unit: enumeration of *Length*\(^{166}\) (optional)
- workingDistance.value: double (optional)

**ObjectiveAnnotationLink**

Used by: *Objective.annotationLinks*

**Properties:**

- child: *Annotation*, see ILink\(^{167}\)
- details.creationEvent: *Event*
- details.externalInfo: *ExternalInfo* (optional)
- details.group: *ExperimenterGroup*
- details.owner: *Experimenter*
- details.updateEvent: *Event*
- parent: *Objective*, see ILink\(^{168}\)
- version: integer (optional), see IMutable\(^{169}\)

\(^{164}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
\(^{165}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
\(^{166}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
\(^{167}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\(^{168}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\(^{169}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
ObjectiveSettings

Used by: Image.objectiveSettings

Properties:

- correctionCollar: double (optional)
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- medium: Medium (optional)
- objective: Objective
- refractiveIndex: double (optional)
- version: integer (optional), see IMutable\textsuperscript{170}

OriginalFile

Used by: FileAnnotation.file, FilesetEntry.originalFile, JobOriginalFileLink.child, OriginalFileAnnotationLink.parent, PixelsOriginalFileMap.parent, Roi.source

Properties:

- annotationLinks: OriginalFileAnnotationLink (multiple)
- atime: timestamp (optional)
- ctime: timestamp (optional)
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- filesetEntries: FilesetEntry (multiple)
- hash: string (optional)
- hasher: ChecksumAlgorithm (optional)
- mimetype: string (optional)
- mtime: timestamp (optional)
- name: string
- path: text
- pixelsFileMaps: PixelsOriginalFileMap (multiple)
- size: long (optional)
- version: integer (optional), see IMutable\textsuperscript{171}

OriginalFileAnnotationLink

Used by: OriginalFile.annotationLinks

Properties:

- child: Annotation, see ILink\textsuperscript{172}
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter

\textsuperscript{170}http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
\textsuperscript{171}http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
\textsuperscript{172}http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
details.updateEvent: Event
parent: OriginalFile, see ILink
version: integer (optional), see IMutable

ParseJob

Properties:

details.creationEvent: Event from Job
details.externalInfo: ExternalInfo (optional) from Job
details.group: ExperimenterGroup from Job
details.owner: Experimenter from Job
details.updateEvent: Event from Job
finished: timestamp (optional) from Job
groupname: string from Job
message: string from Job
originalFileLinks: JobOriginalFileLink (multiple) from Job
params: binary (optional)
scheduledFor: timestamp from Job
started: timestamp (optional) from Job
status: JobStatus from Job
submitted: timestamp from Job
type: string from Job
username: string from Job
version: integer (optional) from Job

Path

Properties:

annotationLinks: ShapeAnnotationLink (multiple) from Shape
d: text (optional)
details.creationEvent: Event from Shape
details.externalInfo: ExternalInfo (optional) from Shape
details.group: ExperimenterGroup from Shape
details.owner: Experimenter from Shape
details.updateEvent: Event from Shape
fillColor: integer (optional) from Shape
fillRule: string (optional) from Shape
fontFamily: string (optional) from Shape
fontSize.unit: enumeration of Length (optional) from Shape
fontSize.value: double (optional) from Shape
fontStretch: string (optional) from Shape
fontStyle: string (optional) from Shape
fontVariant: string (optional) from Shape
fontWeight: string (optional) from Shape
g: string (optional) from Shape
locked: boolean (optional) from Shape
roi: Roi from Shape
strokeColor: integer (optional) from Shape
strokeDashArray: string (optional) from Shape

173 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
175 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
strokeDashOffset: integer (optional) from Shape
strokeLineCap: string (optional) from Shape
strokeLineJoin: string (optional) from Shape
strokeMiterLimit: integer (optional) from Shape
strokeWidth.unit: enumeration of Length\textsuperscript{176} (optional) from Shape
strokeWidth.value: double (optional) from Shape
textValue: text (optional)
theC: integer (optional) from Shape
theT: integer (optional) from Shape
theZ: integer (optional) from Shape
transform: string (optional) from Shape
vectorEffect: string (optional) from Shape
version: integer (optional) from Shape
visibility: boolean (optional) from Shape

**PhotometricInterpretation**

Used by: LogicalChannel.photometricInterpretation

Properties:

details.externalInfo: ExternalInfo (optional)
value: string, see IEnum\textsuperscript{177}

**PixelDataJob**

Properties:

details.creationEvent: Event from Job
details.externalInfo: ExternalInfo (optional) from Job
details.group: ExperimenterGroup from Job
details.owner: Experimenter from Job
details.updateEvent: Event from Job
finished: timestamp (optional) from Job
groupname: string from Job
message: string from Job
originalFileLinks: JobOriginalFileLink (multiple) from Job
scheduledFor: timestamp from Job
started: timestamp (optional) from Job
status: JobStatus from Job
submitted: timestamp from Job
type: string from Job
username: string from Job
version: integer (optional) from Job

**Pixels**

Used by: Channel.pixels, Image.pixels, Mask.pixels, Pixels.relatedTo, PixelsOriginalFileMap.child, PlaneInfo.pixels, RenderingDef.pixels, Thumbnail.pixels

Properties:

channels: Channel (multiple)
details.creationEvent: Event

\textsuperscript{176}\url{http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html}
\textsuperscript{177}\url{http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IEnum.html}
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
dimensionOrder: DimensionOrder
image: Image
methodology: string (optional)
physicalSizeX.unit: enumeration of Length\(^{178}\) (optional)
physicalSizeX.value: double (optional)
physicalSizeY.unit: enumeration of Length\(^{179}\) (optional)
physicalSizeY.value: double (optional)
physicalSizeZ.unit: enumeration of Length\(^{180}\) (optional)
physicalSizeZ.value: double (optional)
pixelsFileMaps: PixelsOriginalFileMap (multiple)
pixelsType: PixelsType
planeInfo: PlaneInfo (multiple)
relatedTo: Pixels (optional)
settings: RenderingDef (multiple)
sha1: string
significantBits: integer (optional)
sizeC: integer
sizeT: integer
sizeX: integer
sizeY: integer
sizeZ: integer
thumbnails: Thumbnail (multiple)
timeIncrement.unit: enumeration of Time\(^{181}\) (optional)
timeIncrement.value: double (optional)
version: integer (optional), see IMutable\(^{182}\)
waveIncrement: integer (optional)
waveStart: integer (optional)

**PixelsOriginalFileMap**

*Used by: OriginalFile.pixelsFileMaps, Pixels.pixelsFileMaps*

**Properties:**

- child: Pixels, see ILink\(^{183}\)
details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
parent: OriginalFile, see ILink\(^{184}\)
version: integer (optional), see IMutable\(^{185}\)

\(^{178}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
\(^{179}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
\(^{180}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
\(^{181}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Time.html
\(^{182}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
\(^{183}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\(^{184}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\(^{185}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
PixelsType

Used by: OTF.pixelsType, Pixels.pixelsType

Properties:

- bitSize: integer (optional)
- details.externalInfo: ExternalInfo (optional)
- value: string, see IEnum

PlaneInfo

Used by: Pixels.planeInfo, PlaneInfoAnnotationLink.parent

Properties:

- annotationLinks: PlaneInfoAnnotationLink (multiple)
- deltaT.unit: enumeration of Time (optional)
- deltaT.value: double (optional)
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- exposureTime.unit: enumeration of Time (optional)
- exposureTime.value: double (optional)
- pixels: Pixels
- positionX.unit: enumeration of Length (optional)
- positionX.value: double (optional)
- positionY.unit: enumeration of Length (optional)
- positionY.value: double (optional)
- positionZ.unit: enumeration of Length (optional)
- positionZ.value: double (optional)
- theC: integer
- theT: integer
- theZ: integer
- version: integer (optional), see IMutable

PlaneInfoAnnotationLink

Used by: PlaneInfo.annotationLinks

Properties:

- child: Annotation, see ILink
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event

---

188 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Time.html
189 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
190 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
191 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
193 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html

17.3. Glossary of all OMERO Model Objects 561
parent: `PlaneInfo`, see `ILink`[^194]
version: `integer (optional)`, see `IMutable`[^195]

### PlaneSlicingContext

**Properties:**
- constant: `boolean`
- `details.creationEvent`: `Event` from `CodomainMapContext`
- `details.externalInfo`: `ExternalInfo (optional)` from `CodomainMapContext`
- `details.group`: `ExperimenterGroup` from `CodomainMapContext`
- `details.owner`: `Experimenter` from `CodomainMapContext`
- `details.updateEvent`: `Event` from `CodomainMapContext`
- `lowerLimit: integer`
- `planePrevious: integer`
- `planeSelected: integer`
- `renderingDef`: `RenderingDef` from `CodomainMapContext`
- `upperLimit: integer`
- version: `integer (optional)` from `CodomainMapContext`

### Plate

**Used by:** `PlateAcquisition.plate`, `PlateAnnotationLink.parent`, `ScreenPlateLink.child`, `Well.plate`

**Properties:**
- `annotationLinks`: `PlateAnnotationLink` (multiple)
- `columnNamingConvention`: `string (optional)`
- `columns`: `integer (optional)`
- `defaultSample`: `integer (optional)`
- `description`: `text (optional)`
- `details.creationEvent`: `Event`
- `details.externalInfo`: `ExternalInfo (optional)`
- `details.group`: `ExperimenterGroup`
- `details.owner`: `Experimenter`
- `details.updateEvent`: `Event`
- `externalIdentifier`: `string (optional)`
- `name`: `string`
- `plateAcquisitions`: `PlateAcquisition` (multiple)
- `rowNamingConvention`: `string (optional)`
- `rows`: `integer (optional)`
- `screenLinks`: `ScreenPlateLink` (multiple)
- `status`: `string (optional)`
- version: `integer (optional)`, see `IMutable`[^196]
- `wellOriginX.unit`: enumeration of `Length`[^197] (optional)
- `wellOriginX.value`: `double (optional)`
- `wellOriginY.unit`: enumeration of `Length`[^198] (optional)
- `wellOriginY.value`: `double (optional)`
- `wells`: `Well` (multiple)

[^194]: http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
[^197]: http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
[^198]: http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
PlateAcquisition

Used by: `Plate.plateAcquisitions`, `PlateAcquisitionAnnotationLink.parent`, `WellSample.plateAcquisition`

Properties:

- annotationLinks: `PlateAcquisitionAnnotationLink` (multiple)
- description: `text` (optional)
- details.creationEvent: `Event`
- details.externalInfo: `ExternalInfo` (optional)
- details.group: `ExperimenterGroup`
- details.owner: `Experimenter`
- details.updateEvent: `Event`
- endTime: `timestamp` (optional)
- maximumFieldCount: `integer` (optional)
- name: `string` (optional)
- plate: `Plate`
- startTime: `timestamp` (optional)
- version: `integer` (optional), see `IMutable`\(^{199}\)
- wellSample: `WellSample` (multiple)

PlateAcquisitionAnnotationLink

Used by: `PlateAcquisition.annotationLinks`

Properties:

- child: `Annotation`, see `ILink`\(^{200}\)
- details.creationEvent: `Event`
- details.externalInfo: `ExternalInfo` (optional)
- details.group: `ExperimenterGroup`
- details.owner: `Experimenter`
- details.updateEvent: `Event`
- parent: `PlateAcquisition`, see `ILink`\(^{201}\)
- version: `integer` (optional), see `IMutable`\(^{202}\)

PlateAnnotationLink

Used by: `Plate.annotationLinks`

Properties:

- child: `Annotation`, see `ILink`\(^{203}\)
- details.creationEvent: `Event`
- details.externalInfo: `ExternalInfo` (optional)
- details.group: `ExperimenterGroup`
- details.owner: `Experimenter`
- details.updateEvent: `Event`
- parent: `Plate`, see `ILink`\(^{204}\)
- version: `integer` (optional), see `IMutable`\(^{205}\)

\(^{199}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html

\(^{200}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html

\(^{201}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html

\(^{202}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html

\(^{203}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html

\(^{204}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html

\(^{205}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
### Point

**Properties:**

- `annotationLinks`: `ShapeAnnotationLink (multiple) from Shape`
- `cx`: `double (optional)`
- `cy`: `double (optional)`
- `details.creationEvent`: `Event from Shape`
- `details.externalInfo`: `ExternalInfo (optional) from Shape`
- `details.group`: `ExperimenterGroup from Shape`
- `details.owner`: `Experimenter from Shape`
- `details.updateEvent`: `Event from Shape`
- `fillColor`: `integer (optional) from Shape`
- `fillRule`: `string (optional) from Shape`
- `fontFamily`: `string (optional) from Shape`
- `fontSize.unit`: enumeration of `Length`\(^206\) (optional) from `Shape`
- `fontSize.value`: `double (optional) from Shape`
- `fontStretch`: `string (optional) from Shape`
- `fontStyle`: `string (optional) from Shape`
- `fontVariant`: `string (optional) from Shape`
- `fontWeight`: `string (optional) from Shape`
- `g`: `string (optional) from Shape`
- `locked`: `boolean (optional) from Shape`
- `roi`: `Roi from Shape`
- `strokeColor`: `integer (optional) from Shape`
- `strokeDashArray`: `string (optional) from Shape`
- `strokeDashOffset`: `integer (optional) from Shape`
- `strokeLineCap`: `string (optional) from Shape`
- `strokeLineJoin`: `string (optional) from Shape`
- `strokeMiterLimit`: `integer (optional) from Shape`
- `strokeWidth.unit`: enumeration of `Length`\(^207\) (optional) from `Shape`
- `strokeWidth.value`: `double (optional) from Shape`
- `textValue`: `text (optional)`
- `theC`: `integer (optional) from Shape`
- `theT`: `integer (optional) from Shape`
- `theZ`: `integer (optional) from Shape`
- `transform`: `string (optional) from Shape`
- `vectorEffect`: `string (optional) from Shape`
- `version`: `integer (optional) from Shape`
- `visibility`: `boolean (optional) from Shape`

### Polygon

**Properties:**

- `annotationLinks`: `ShapeAnnotationLink (multiple) from Shape`
- `details.creationEvent`: `Event from Shape`
- `details.externalInfo`: `ExternalInfo (optional) from Shape`
- `details.group`: `ExperimenterGroup from Shape`
- `details.owner`: `Experimenter from Shape`
- `details.updateEvent`: `Event from Shape`
- `fillColor`: `integer (optional) from Shape`

---

\(^{206}\) http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html

\(^{207}\) http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
fillRule: string (optional) from Shape
fontFamily: string (optional) from Shape
fontSize.unit: enumeration of Length\(^{208}\) (optional) from Shape
fontSize.value: double (optional) from Shape
fontStretch: string (optional) from Shape
fontStyle: string (optional) from Shape
fontVariant: string (optional) from Shape
fontWeight: string (optional) from Shape
g: string (optional) from Shape
locked: boolean (optional) from Shape
points: text (optional)
roi: Roi from Shape
strokeColor: integer (optional) from Shape
strokeDashArray: string (optional) from Shape
strokeDashOffset: integer (optional) from Shape
strokeLineCap: string (optional) from Shape
strokeLineJoin: string (optional) from Shape
strokeMiterLimit: integer (optional) from Shape
strokeWidth.unit: enumeration of Length\(^{209}\) (optional) from Shape
strokeWidth.value: double (optional) from Shape
textValue: text (optional)
theC: integer (optional) from Shape
theT: integer (optional) from Shape
theZ: integer (optional) from Shape
transform: string (optional) from Shape
vectorEffect: string (optional) from Shape
version: integer (optional) from Shape
visibility: boolean (optional) from Shape

Polyline

Properties:

annotationLinks: ShapeAnnotationLink (multiple) from Shape
details.creationEvent: Event from Shape
details.externalInfo: ExternalInfo (optional) from Shape
details.group: ExperimenterGroup from Shape
details.owner: Experimenter from Shape
details.updateEvent: Event from Shape
fillColor: integer (optional) from Shape
fillRule: string (optional) from Shape
fontName: string (optional) from Shape
fontSize.unit: enumeration of Length\(^{210}\) (optional) from Shape
fontSize.value: double (optional) from Shape
fontStretch: string (optional) from Shape
fontStyle: string (optional) from Shape
fontVariant: string (optional) from Shape
fontWeight: string (optional) from Shape
g: string (optional) from Shape
locked: boolean (optional) from Shape
points: text (optional)

---

\(^{208}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html

\(^{209}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html

\(^{210}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
roi: Roi from Shape
strokeColor: integer (optional) from Shape
strokeDashArray: string (optional) from Shape
strokeDashOffset: integer (optional) from Shape
strokeLineCap: string (optional) from Shape
strokeLineJoin: string (optional) from Shape
strokeMiterLimit: integer (optional) from Shape
strokeWidth.unit: enumeration of Length\(^{211}\) (optional) from Shape
strokeWidth.value: double (optional) from Shape
textValue: text (optional)
theC: integer (optional) from Shape
theT: integer (optional) from Shape
theZ: integer (optional) from Shape
transform: string (optional) from Shape
vectorEffect: string (optional) from Shape
version: integer (optional) from Shape
visibility: boolean (optional) from Shape

Project

Used by: ProjectAnnotationLink.parent, ProjectDatasetLink.parent

Properties:

annotationLinks: ProjectAnnotationLink (multiple)
datasetLinks: ProjectDatasetLink (multiple)
description: text (optional)
details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
name: string
version: integer (optional), see IMutable\(^{212}\)

ProjectAnnotationLink

Used by: Project.annotationLinks

Properties:

child: Annotation, see ILink\(^{213}\)
details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
parent: Project, see ILink\(^{214}\)
version: integer (optional), see IMutable\(^{215}\)

---

\(^{211}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
\(^{212}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
\(^{213}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\(^{214}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\(^{215}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
ProjectDatasetLink

Used by: Dataset.projectLinks, Project.datasetLinks

Properties:
- child: Dataset, see ILink
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- parent: Project, see ILink
- version: integer (optional), see IMutable

Pulse

Used by: Laser.pulse

Properties:
- details.externalInfo: ExternalInfo (optional)
- value: string, see IEnum

QuantumDef

Used by: RenderingDef.quantization

Properties:
- bitResolution: integer
- cdEnd: integer
- cdStart: integer
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- version: integer (optional), see IMutable

Reagent

Used by: ReagentAnnotationLink.parent, Screen.reagents, WellReagentLink.child

Properties:
- annotationLinks: ReagentAnnotationLink (multiple)
- description: text (optional)
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- name: string (optional)
reagentIdentifier: string (optional)

screen: Screen

version: integer (optional), see IMutable

wellLinks: WellReagentLink (multiple)

**ReagentAnnotationLink**

Used by: Reagent.annotationLinks

**Properties:**

child: Annotation, see ILink

details.creationEvent: Event

details.externalInfo: ExternalInfo (optional)

details.group: ExperimenterGroup

details.owner: Experimenter

details.updateEvent: Event

parent: Reagent, see ILink

version: integer (optional), see IMutable

**Rectangle**

**Properties:**

annotationLinks: ShapeAnnotationLink (multiple) from Shape

details.creationEvent: Event from Shape

details.externalInfo: ExternalInfo (optional) from Shape

details.group: ExperimenterGroup from Shape

details.owner: Experimenter from Shape

details.updateEvent: Event from Shape

fillColor: integer (optional) from Shape

fillRule: string (optional) from Shape

fontFamily: string (optional) from Shape

fontSize.unit: enumeration of Length (optional) from Shape

fontSize.value: double (optional) from Shape

fontStretch: string (optional) from Shape

fontStyle: string (optional) from Shape

fontVariant: string (optional) from Shape

fontWeight: string (optional) from Shape

g: string (optional) from Shape

height: double (optional)

locked: boolean (optional) from Shape

roi: Roi from Shape

rx: double (optional)

strokeColor: integer (optional) from Shape

strokeDashArray: string (optional) from Shape

strokeDashOffset: integer (optional) from Shape

strokeLineCap: string (optional) from Shape

strokeLineJoin: string (optional) from Shape

strokeMiterLimit: integer (optional) from Shape

---

221 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
222 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
223 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
224 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
225 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html

17.3. Glossary of all OMERO Model Objects 568
strokeWidth.unit: enumeration of Length (optional) from Shape
strokeWidth.value: double (optional) from Shape
textValue: text (optional)
theC: integer (optional) from Shape
theT: integer (optional) from Shape
theZ: integer (optional) from Shape
transform: string (optional) from Shape
vectorEffect: string (optional) from Shape
version: integer (optional) from Shape
visibility: boolean (optional) from Shape
width: double (optional)
x: double (optional)
y: double (optional)

RenderingDef

Used by: ChannelBinding.renderingDef, CodomainMapContext.renderingDef, Pixels.settings

Properties:
  compression: double (optional)
defaultT: integer
defaultZ: integer
details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
model: RenderingModel
name: string (optional)
pixels: Pixels
quantization: QuantumDef
spatialDomainEnhancement: CodomainMapContext (multiple)
version: integer (optional), see IMutable
waveRendering: ChannelBinding (multiple)

RenderingModel

Used by: RenderingDef.model

Properties:
  details.externalInfo: ExternalInfo (optional)
  value: string, see IEnum

ReverseIntensityContext

Properties:
  details.creationEvent: Event from CodomainMapContext
details.externalInfo: ExternalInfo (optional) from CodomainMapContext
details.group: ExperimenterGroup from CodomainMapContext
details.owner: Experimenter from CodomainMapContext

---

226 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
228 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IEnum.html
details.updateEvent: Event from CodomainMapContext
renderingDef: RenderingDef from CodomainMapContext
reverse: boolean
version: integer (optional) from CodomainMapContext

Roi

Used by: Image.rois, RoiAnnotationLink.parent, Shape.roi

Properties:

annotationLinks: RoiAnnotationLink (multiple)
description: text (optional)
details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
image: Image (optional)
keywords: list (optional)
name: string (optional)
namespaces: list (optional)
shapes: Shape (multiple)
source: OriginalFile (optional)
version: integer (optional), see IMutable

RoiAnnotationLink

Used by: Roi.annotationLinks

Properties:

child: Annotation, see ILink

details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
details.owner: Roi, see ILink
parent: Roi, see ILink
version: integer (optional), see IMutable

Screen

Used by: Reagent.screen, ScreenAnnotationLink.parent, ScreenPlateLink.parent

Properties:

annotationLinks: ScreenAnnotationLink (multiple)
description: text (optional)
details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter

http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html

17.3. Glossary of all OMERO Model Objects
details.updateEvent: Event
name: string
plateLinks: ScreenPlateLink (multiple)
protocolDescription: string (optional)
protocolIdentifier: string (optional)
reagentSetDescription: string (optional)
reagentSetIdentifier: string (optional)
reagents: Reagent (multiple)
type: string (optional)
version: integer (optional), see IMutable.

ScreenAnnotationLink

Used by: Screen.annotationLinks

Properties:
child: Annotation, see ILink.
details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
parent: Screen, see ILink.
version: integer (optional), see IMutable.

ScreenPlateLink

Used by: Plate.screenLinks, Screen.plateLinks

Properties:
child: Plate, see ILink.
details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
parent: Screen, see ILink.
version: integer (optional), see IMutable.

ScriptJob

Properties:
description: string (optional)
details.creationEvent: Event from Job
details.externalInfo: ExternalInfo (optional) from Job
details.group: ExperimenterGroup from Job
details.owner: Experimenter from Job

234 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
235 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
236 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
237 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
238 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
239 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
details.updateEvent: Event from Job
finished: timestamp (optional) from Job
groupname: string from Job
message: string from Job
originalFileLinks: JobOriginalFileLink (multiple) from Job
scheduledFor: timestamp from Job
started: timestamp (optional) from Job
status: JobStatus from Job
submitted: timestamp from Job
type: string from Job
username: string from Job
version: integer (optional) from Job

Session
Subclasses: Share
Used by: Event.session, Node.sessions, SessionAnnotationLink.parent

Properties:
annotationLinks: SessionAnnotationLink (multiple)
closed: timestamp (optional)
defaultEventType: string
details.externalInfo: ExternalInfo (optional)
events: Event (multiple)
message: text (optional)
node: Node
owner: Experimenter
started: timestamp
timeToIdle: long
timeToLive: long
userAgent: string (optional)
userIP: string (optional)
uuid: string
version: integer (optional), see IMutable

SessionAnnotationLink
Used by: Session.annotationLinks

Properties:
child: Annotation, see ILink

details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
parent: Session, see ILink
version: integer (optional), see IMutable

17.3. Glossary of all OMERO Model Objects

http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
Shape

Subclasses: Ellipse, Label, Line, Mask, Path, Point, Polygon, Polyline, Rectangle

Used by: Roi.shapes, ShapeAnnotationLink.parent

Properties:

- annotationLinks: ShapeAnnotationLink (multiple)
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- fillColor: integer (optional)
- fillRule: string (optional)
- fontFamily: string (optional)
- fontSize.unit: enumeration of Length
- fontSize.value: double (optional)
- fontStyle: string (optional)
- fontWeight: string (optional)
- g: string (optional)
- locked: boolean (optional)
- roi: Roi
- strokeColor: integer (optional)
- strokeDashArray: string (optional)
- strokeDashOffset: integer (optional)
- strokeLineCap: string (optional)
- strokeLineJoin: string (optional)
- strokeMiterLimit: integer (optional)
- strokeWidth.unit: enumeration of Length
- strokeWidth.value: double (optional)
- theC: integer (optional)
- theT: integer (optional)
- theZ: integer (optional)
- transform: string (optional)
- vectorEffect: string (optional)
- version: integer (optional), see IMutable
- visibility: boolean (optional)

ShapeAnnotationLink

Used by: Shape.annotationLinks

Properties:

- child: Annotation, see ILink
- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter

---

244 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
245 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
246 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
247 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html

17.3. Glossary of all OMERO Model Objects 573
details.updateEvent: Event
parent: Shape, see ILink\(^{248}\)
version: integer (optional), see IMutable\(^{249}\)

Share

Used by: ShareMember.parent

Properties:

- active: boolean
- annotationLinks: SessionAnnotationLink (multiple) from Session
- closed: timestamp (optional) from Session
- data: binary
- defaultEventType: string from Session
- details.externalInfo: ExternalInfo (optional) from Session
- events: Event (multiple) from Session
- group: ExperimenterGroup
- itemCount: long
- message: text (optional) from Session
- node: Node from Session
- owner: Experimenter from Session
- started: timestamp from Session
- timeToIdle: long from Session
- timeToLive: long from Session
- userAgent: string (optional) from Session
- userIP: string (optional) from Session
- uuid: string from Session
- version: integer (optional) from Session

ShareMember

Properties:

- child: Experimenter, see ILink\(^{250}\)
- details.externalInfo: ExternalInfo (optional)
- parent: Share, see ILink\(^{251}\)
- version: integer (optional), see IMutable\(^{252}\)

StageLabel

Used by: Image.stageLabel

Properties:

- details.creationEvent: Event
- details.externalInfo: ExternalInfo (optional)
- details.group: ExperimenterGroup
- details.owner: Experimenter
- details.updateEvent: Event
- name: string

\(^{248}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\(^{249}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
\(^{250}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\(^{251}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
\(^{252}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
positionX.unit: enumeration of Length (optional)
positionX.value: double (optional)
positionY.unit: enumeration of Length (optional)
positionY.value: double (optional)
positionZ.unit: enumeration of Length (optional)
positionZ.value: double (optional)
version: integer (optional), see IMutable

StatsInfo

Used by: Channel.statsInfo

Properties:

details.creationEvent: Event

details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
globalMax: double
globalMin: double
version: integer (optional), see IMutable

TagAnnotation

Properties:

annotationLinks: AnnotationAnnotationLink (multiple) from Annotation
description: text (optional) from Annotation
details.creationEvent: Event from Annotation
details.externalInfo: ExternalInfo (optional) from Annotation
details.group: ExperimenterGroup from Annotation
details.owner: Experimenter from Annotation
details.updateEvent: Event from Annotation
name: string (optional) from Annotation
ns: string (optional) from Annotation
textValue: text (optional) from TextAnnotation
version: integer (optional) from Annotation

TermAnnotation

Properties:

annotationLinks: AnnotationAnnotationLink (multiple) from Annotation
description: text (optional) from Annotation
details.creationEvent: Event from Annotation
details.externalInfo: ExternalInfo (optional) from Annotation
details.group: ExperimenterGroup from Annotation
details.owner: Experimenter from Annotation
details.updateEvent: Event from Annotation
name: string (optional) from Annotation

253 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
254 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
255 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
256 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
257 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
ns: string (optional) from Annotation
termValue: text (optional)
version: integer (optional) from Annotation

**TextAnnotation**

Subclasses: *CommentAnnotation, TagAnnotation, XmlAnnotation*

**Properties:**

- annotationLinks: AnnotationAnnotationLink (multiple) from Annotation
description: text (optional) from Annotation
details.creationEvent: Event from Annotation
details.externalInfo: ExternalInfo (optional) from Annotation
details.group: ExperimenterGroup from Annotation
details.owner: Experimenter from Annotation
details.updateEvent: Event from Annotation
name: string (optional) from Annotation
ns: string (optional) from Annotation
textValue: text (optional)
version: integer (optional) from Annotation

**Thumbnail**

Used by: Pixelsthumbnails

**Properties:**

details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
mimeType: string
pixels: Pixels
ref: string (optional)
sizeX: integer
sizeY: integer
version: integer (optional), see IMutable

**ThumbnailGenerationJob**

**Properties:**

details.creationEvent: Event from Job
details.externalInfo: ExternalInfo (optional) from Job
details.group: ExperimenterGroup from Job
details.owner: Experimenter from Job
details.updateEvent: Event from Job
finished: timestamp (optional) from Job
groupname: string from Job
message: string from Job
originalFileLinks: JobOriginalFileLink (multiple) from Job
scheduledFor: timestamp from Job
started: timestamp (optional) from Job

http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html

17.3. Glossary of all OMERO Model Objects
status: JobStatus from Job
submitted: timestamp from Job
type: string from Job
username: string from Job
version: integer (optional) from Job

TimestampAnnotation

Properties:
annotationLinks: AnnotationAnnotationLink (multiple) from Annotation
description: text (optional) from Annotation
details.creationEvent: Event from Annotation
details.externalInfo: ExternalInfo (optional) from Annotation
details.group: ExperimenterGroup from Annotation
details.owner: Experimenter from Annotation
details.updateEvent: Event from Annotation
name: string (optional) from Annotation
ns: string (optional) from Annotation
timeValue: timestamp (optional)
version: integer (optional) from Annotation

TransmittanceRange

Used by: Filter.transmittanceRange

Properties:
cutIn.unit: enumeration of Length\(^{259}\) (optional)
cutIn.value: double (optional)
cutInTolerance.unit: enumeration of Length\(^{260}\) (optional)
cutInTolerance.value: double (optional)
cutOut.unit: enumeration of Length\(^{261}\) (optional)
cutOut.value: double (optional)
cutOutTolerance.unit: enumeration of Length\(^{262}\) (optional)
cutOutTolerance.value: double (optional)
details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
transmittance: double (optional)
version: integer (optional), see IMutable\(^{263}\)

TypeAnnotation

Subclasses: FileAnnotation

Properties:
annotationLinks: AnnotationAnnotationLink (multiple) from Annotation
description: text (optional) from Annotation

\(^{259}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
\(^{260}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
\(^{261}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
\(^{262}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
\(^{263}\)http://downloads.openmicroscopy.org/latest/omero/api/ome/model/IMutable.html
**UploadJob**

**Properties:**

- **details.creationEvent:** *Event* from *Job*
- **details.externalInfo:** *ExternalInfo* (optional) from *Job*
- **details.group:** *ExperimenterGroup* from *Job*
- **details.owner:** *Experimenter* from *Job*
- **details.updateEvent:** *Event* from *Job*
- **finished:** timestamp (optional) from *Job*
- **groupname:** string from *Job*
- **message:** string from *Job*
- **originalFileLinks:** *JobOriginalFileLink* (multiple) from *Job*
- **scheduledFor:** timestamp from *Job*
- **started:** timestamp (optional) from *Job*
- **status:** *JobStatus* from *Job*
- **submitted:** timestamp from *Job*
- **type:** string from *Job*
- **username:** string from *Job*
- **version:** integer (optional) from *Job*
- **versionInfo:** list (multiple)

**Well**

Used by: *Plate.wells*, *WellAnnotationLink.parent*, *WellReagentLink.parent*, *WellSample.well*

**Properties:**

- **alpha:** integer (optional)
- **annotationLinks:** *WellAnnotationLink* (multiple)
- **blue:** integer (optional)
- **column:** integer (optional)
- **details.creationEvent:** *Event*
- **details.externalInfo:** *ExternalInfo* (optional)
- **details.group:** *ExperimenterGroup*
- **details.owner:** *Experimenter*
- **details.updateEvent:** *Event*
- **externalDescription:** string (optional)
- **externalIdentifier:** string (optional)
- **green:** integer (optional)
- **plate:** *Plate*
- **reagentLinks:** *WellReagentLink* (multiple)
- **red:** integer (optional)
- **row:** integer (optional)
- **status:** string (optional)
- **type:** string (optional)
version: integer (optional), see IMutable
wellSamples: WellSample (multiple)

WellAnnotationLink

Used by: Well.annotationLinks

Properties:

- child: Annotation, see ILink
details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
parent: Well, see ILink
version: integer (optional), see IMutable

WellReagentLink

Used by: Reagent.wellLinks, Well.reagentLinks

Properties:

- child: Reagent, see ILink
details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
parent: Well, see ILink
version: integer (optional), see IMutable

WellSample

Used by: Image.wellSamples, PlateAcquisition.wellSample, Well.wellSamples

Properties:

- details.creationEvent: Event
details.externalInfo: ExternalInfo (optional)
details.group: ExperimenterGroup
details.owner: Experimenter
details.updateEvent: Event
image: Image
plateAcquisition: PlateAcquisition (optional)
posX.unit: enumeration of Length
posX.value: double (optional)
posY.unit: enumeration of Length
posY.value: double (optional)

---

265 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
266 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
267 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
268 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
269 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
270 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/ILink.html
271 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
272 http://downloads.openmicroscopy.org/latest/omero/api/ome/model/units/Length.html
timepoint: timestamp (optional)
version: integer (optional), see IMutable
well: Well

XmlAnnotation

Properties:

- annotationLinks: AnnotationAnnotationLink (multiple) from Annotation
- description: text (optional) from Annotation
- details.creationEvent: Event from Annotation
- details.externalInfo: ExternalInfo (optional) from Annotation
- details.group: ExperimenterGroup from Annotation
- details.owner: Experimenter from Annotation
- details.updateEvent: Event from Annotation
- name: string (optional) from Annotation
- ns: string (optional) from Annotation
- textValue: text (optional) from TextAnnotation
- version: integer (optional) from Annotation

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

17.4 Units

A number of properties in the OME model are physical measurements which inherently have a unit associated with them. Earlier versions of OME defined a default unit for the measurement. Now users, clients, and acquisition systems can specify the unit themselves rather than converting their internal unit to the OME default.

See also:
Data Model documentation for units

17.4.1 Supported units

- Electric potential
- Frequency
- Length
- Power
- Pressure
- Temperature
- Time

Each of the supported units contain a number of values from the International System of Units (SI) in the most common prefixes from yotta (10^24) to yocto (10^-24). The Length unit also contains values from the Imperial system as well as internal values which are internal to OME: reference frame and pixel.

274 http://www.openmicroscopy.org/site/support/ome-model/developers/ome-units.html
276 http://downloads.openmicroscopy.org/latest/omero/api/omero/model/FrequencyI.html
277 http://downloads.openmicroscopy.org/latest/omero/api/omero/model/LengthI.html
278 http://downloads.openmicroscopy.org/latest/omero/api/omero/model/PowerI.html
279 http://downloads.openmicroscopy.org/latest/omero/api/omero/model/PressureI.html

17.4. Units
17.4.2 Unit fields

The following fields in the OMERO model have a unit type:

<table>
<thead>
<tr>
<th>Class</th>
<th>Field</th>
<th>Type</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel</td>
<td>EmissionWavelength</td>
<td>Length</td>
<td>nm</td>
</tr>
<tr>
<td>Channel</td>
<td>ExcitationWavelength</td>
<td>Length</td>
<td>nm</td>
</tr>
<tr>
<td>Channel</td>
<td>PinholeSize</td>
<td>Length</td>
<td>µm</td>
</tr>
<tr>
<td>Detector</td>
<td>Voltage</td>
<td>ElectricPotential</td>
<td>V</td>
</tr>
<tr>
<td>DetectorSettings</td>
<td>ReadOutRate</td>
<td>Frequency</td>
<td>MHz</td>
</tr>
<tr>
<td>DetectorSettings</td>
<td>Voltage</td>
<td>ElectricPotential</td>
<td>V</td>
</tr>
<tr>
<td>ImagingEnvironment</td>
<td>AirPressure</td>
<td>Pressure</td>
<td>mbar</td>
</tr>
<tr>
<td>ImagingEnvironment</td>
<td>Temperature</td>
<td>Temperature</td>
<td>°C</td>
</tr>
<tr>
<td>Laser</td>
<td>RepetitionRate</td>
<td>Frequency</td>
<td>Hz</td>
</tr>
<tr>
<td>Laser</td>
<td>Wavelength</td>
<td>Length</td>
<td>nm</td>
</tr>
<tr>
<td>LightSource</td>
<td>Power</td>
<td>Power</td>
<td>mW</td>
</tr>
<tr>
<td>LightSourceSettings</td>
<td>Wavelength</td>
<td>Length</td>
<td>nm</td>
</tr>
<tr>
<td>Objective</td>
<td>WorkingDistance</td>
<td>Length</td>
<td>µm</td>
</tr>
<tr>
<td>Pixels</td>
<td>PhysicalSizeX</td>
<td>Length</td>
<td>µm</td>
</tr>
<tr>
<td>Pixels</td>
<td>PhysicalSizeY</td>
<td>Length</td>
<td>µm</td>
</tr>
<tr>
<td>Pixels</td>
<td>PhysicalSizeZ</td>
<td>Length</td>
<td>µm</td>
</tr>
<tr>
<td>Pixels</td>
<td>TimeIncrement</td>
<td>Time</td>
<td>s</td>
</tr>
<tr>
<td>Plane</td>
<td>DeltaT</td>
<td>Time</td>
<td>s</td>
</tr>
<tr>
<td>Plane</td>
<td>ExposureTime</td>
<td>Time</td>
<td>s</td>
</tr>
<tr>
<td>Plane</td>
<td>PositionX</td>
<td>Length</td>
<td>reference frame</td>
</tr>
<tr>
<td>Plane</td>
<td>PositionY</td>
<td>Length</td>
<td>reference frame</td>
</tr>
<tr>
<td>Plane</td>
<td>PositionZ</td>
<td>Length</td>
<td>reference frame</td>
</tr>
<tr>
<td>Plate</td>
<td>WellOriginX</td>
<td>Length</td>
<td>reference frame</td>
</tr>
<tr>
<td>Plate</td>
<td>WellOriginY</td>
<td>Length</td>
<td>reference frame</td>
</tr>
<tr>
<td>Shape</td>
<td>FontSize</td>
<td>Length</td>
<td>pt</td>
</tr>
<tr>
<td>Shape</td>
<td>StrokeWidth</td>
<td>Length</td>
<td>pixel</td>
</tr>
<tr>
<td>StageLabel</td>
<td>X</td>
<td>Length</td>
<td>reference frame</td>
</tr>
<tr>
<td>StageLabel</td>
<td>Y</td>
<td>Length</td>
<td>reference frame</td>
</tr>
<tr>
<td>StageLabel</td>
<td>Z</td>
<td>Length</td>
<td>reference frame</td>
</tr>
<tr>
<td>TransmittanceRange</td>
<td>CutIn</td>
<td>Length</td>
<td>nm</td>
</tr>
<tr>
<td>TransmittanceRange</td>
<td>CutInTolerance</td>
<td>Length</td>
<td>nm</td>
</tr>
<tr>
<td>TransmittanceRange</td>
<td>CutOut</td>
<td>Length</td>
<td>nm</td>
</tr>
<tr>
<td>TransmittanceRange</td>
<td>CutOutTolerance</td>
<td>Length</td>
<td>nm</td>
</tr>
<tr>
<td>WellSample</td>
<td>PositionX</td>
<td>Length</td>
<td>reference frame</td>
</tr>
<tr>
<td>WellSample</td>
<td>PositionY</td>
<td>Length</td>
<td>reference frame</td>
</tr>
</tbody>
</table>

Units of type *reference frame* cannot be assumed comparable to units from other properties including those with type *reference frame*.

17.4.3 Unit objects

Each unit quantity consists of a double-precision scalar and an enumeration which chooses one of the pre-defined values from the model. In code, uppercase spellings of the enumerations are used, while in the schema, in OME-XML files, and in the database, the Unicode symbol for the unit is used.

<table>
<thead>
<tr>
<th>Language</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ice</td>
<td><code>enum UnitsLength { MICROM, ... };</code></td>
</tr>
<tr>
<td>Java and Python</td>
<td><code>omero.model.enums.UnitsLength.MICROM</code></td>
</tr>
<tr>
<td>C++</td>
<td><code>omero::model::enums::MICROM</code></td>
</tr>
<tr>
<td>PostgreSQL</td>
<td><code>'µm'::unitslength</code></td>
</tr>
</tbody>
</table>
Defining a unit

Pixels p = ...; // Defined elsewhere
Length l = new LengthI(2.1, UnitsLength.MICROM); // µm
p.setPhysicalSizeX(l);
p.setPhysicalSizeY(l);
iUpdatePrx.saveObject(p);

The above stores a Pixels object in the database with X and Y physical lengths of “µm”.

Converting a unit

Often a measurement will not be in the most convenient unit for display, e.g. 0.00001 mm. could better be expressed in microns. In order to convert between units, pass the measurement that you have available to a constructor of the same type, passing in the target unit that you would like to see:

Pixels p = ...; // As saved above
Length l1 = p.getPhysicalSizeX(); // 2.1 microns
Length l2 = new LengthI(x1, UnitsLength.NM); // As nanometers

Getting a symbol

The enumerations used in the “units” field of each measurement is of type omero.model.enums.UnitsNAME where NAME is Length, Temperature, etc. These members of that enumeration are all uppercased, code-safe versions of the unit name. To get the symbol as defined in the SI specification, for example, use the getSymbol method:

Length l1 = ...; // As above
l1.getSymbol(); // Returns “µm”

17.4.4 Querying units

In HQL queries, the scalar and the enumeration value can be separately retrieved.

select planeInfo.exposureTime.value from PlaneInfo planeInfo ...

will retrieve just the double scalar value while

select planeInfo.exposureTime.unit from PlaneInfo planeInfo ...

will retrieve a string representation of the enum which can be used in each language to create an enum object, e.g.:

UnitsTime.valueOf(unit); // Java
getattr(UnitsTime, unit) # Python

To load the symbolic representation of the enum which is used internally in the database and is more concise, use an HQL cast:

select cast(planeInfo.exposureTime.unit as text) from PlaneInfo planeInfo ...

Returning the entire unit quantity will result in a hash map with the various representations:
select planeInfo.exposureTime from PlaneInfo planeInfo ...
('symbol': 's', 'unit': 'SECOND', 'value': 1.2000000476837158)

See also:


Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

17.5 Key-value pairs

Ongoing advancements in microscopic techniques, analysis systems, and other areas for which OME attempts to provide metadata exchange require a certain flexibility in the OME model. There is a need to store instrument configuration, script parameters, and similar for later use.

Basic text representations are too difficult to parse to be of significant use. An XML format can be more easily parsed, but a single format would have to be agreed upon by all users. Therefore, it is useful to add particular extension points to the model for which no consensus on a data format has been reached, but where more structure than just text is needed.

A hash map was the most likely candidate for this data; however, rather than limit this to traditional associative arrays, OME maps are defined as a “ordered list of key-value pairs”. The benefit of this representation is that configuration files which are standard for Java java.util.Properties objects can be represented fully in a single map. Duplicates are maintained since there is no unique constraint on the list of pairs.

In most cases, the interpretation of the ordered list will be such that the final value for a particular key “wins” as if each value were placed into a hash map in order, with duplicate values replacing previous ones.

17.5.1 OME-XML

In the OME-XML model, these maps are represented in a compact format. Any map field is defined by the MapPairs complex type which consists of M elements of the form:

\[<M K=\text{"key"} >\text{value}</M>\]

17.5.2 OMERO languages

In OMERO, a slightly more verbose representation of these objects is used. Each map type consists of a list or vector in the respective language, composed of NamedValue objects and possibly nulls.

// OMERO.java
ImagingEnvironment environment = new ImagingEnvironment();
environment.setMap(new List<NamedValue>());
environment.getMap().add(new NamedValue("altitude", "1000m"));
image.setImagingEnvironment(environment);

Fields

The concrete fields which are present in the model are currently:

- ExperimenterGroup.config
• GenericExcitationSource.map
• ImagingEnvironment.map
• ImportJob.versionInfo

More will be added as demand increases.

### 17.5.3 MapAnnotations

In addition to the fields above, there is also a *structured annotation* which contains a key-valued pair, the `MapAnnotation`.

```cpp
// OMERO.cpp
MapAnnotation ann = new MapAnnotationI();
ann->getMapValue().push_back(new NamedValueI("run", "5.0"));
ann->getMapValue().push_back(new NamedValueI("run", "4.9"));
ann->getMapValue().push_back(new NamedValueI("run", "5.1"));
```

This permits the flexible attachment of key-value pairs to any of the OME types which are annotatable. Such annotations attached to key UI elements like images and datasets will be presented by the clients, and can be edited with the appropriate permissions. See Managing Data[^282] on OMERO User Help[^283] for more information.

### 17.5.4 Storage and queries

Each map-based field in the OMERO model is represented by an extra table of the form `$className_$fieldName`. For example, `MapAnnotation.mapValue` becomes `annotation.mapValue`, where the loss of “map” from the class name is due to the subclassing of `Annotation` by `MapAnnotation`.

In general, use of the specific tables is not necessary and it suffices to write HQL queries based on the classes and field names themselves.

#### Find the value for a key

```sql
select nv.value from MapAnnotation ann
join ann.mapValue as nv
where nv.name = 'altitude'
```

#### Finding objects with a key

```sql
select ann from MapAnnotation ann
join ann.mapValue as nv
where nv.name = 'altitude'
```

#### Finding objects without a key

```sql
select ann from MapAnnotation ann
where not exists(
  from MapAnnotation m2
  join m2.mapValue as nv2
  where nv2.name like 'size%')
```

[^282]: http://help.openmicroscopy.org/managing-data.html#keyvalue
[^283]: http://help.openmicroscopy.org/
Finding objects with multiple values

```sql
select ann from MapAnnotation ann
join ann.mapValue as nv1
join ann.mapValue as nv2
where nv1.name = 'date'
and nv2.name = 'owner'
```

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 17.6 Available transformations

<table>
<thead>
<tr>
<th>Available transforms</th>
<th>Direction</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003-FC-to-2007-06.xsl</td>
<td>upgrade</td>
<td>excellent</td>
</tr>
<tr>
<td>2003-FC-to-2008-09.xsl</td>
<td>upgrade</td>
<td>excellent</td>
</tr>
<tr>
<td>2007-06-to-2008-02.xsl</td>
<td>upgrade</td>
<td>excellent</td>
</tr>
<tr>
<td>2007-06-to-2008-09.xsl</td>
<td>upgrade</td>
<td>excellent</td>
</tr>
<tr>
<td>2008-02-to-2008-09.xsl</td>
<td>upgrade</td>
<td>excellent</td>
</tr>
<tr>
<td>2008-09-to-2009-09.xsl</td>
<td>upgrade</td>
<td>excellent</td>
</tr>
<tr>
<td>2009-09-to-2010-04.xsl</td>
<td>upgrade</td>
<td>excellent</td>
</tr>
<tr>
<td>2010-04-to-2010-06.xsl</td>
<td>upgrade</td>
<td>excellent</td>
</tr>
<tr>
<td>2010-06-to-2011-06.xsl</td>
<td>upgrade</td>
<td>excellent</td>
</tr>
<tr>
<td>2011-06-to-2012-06.xsl</td>
<td>upgrade</td>
<td>excellent</td>
</tr>
<tr>
<td>2012-06-to-2013-06.xsl</td>
<td>upgrade</td>
<td>excellent</td>
</tr>
<tr>
<td>2013-06-to-2015-01.xsl</td>
<td>upgrade</td>
<td>excellent</td>
</tr>
<tr>
<td>2010-06-to-2003-FC.xsl</td>
<td>downgrade</td>
<td>poor (very lossy)</td>
</tr>
<tr>
<td>2010-06-to-2008-02.xsl</td>
<td>downgrade</td>
<td>fair (lossy)</td>
</tr>
<tr>
<td>2011-06-to-2010-06.xsl</td>
<td>downgrade</td>
<td>good</td>
</tr>
<tr>
<td>2012-06-to-2011-06.xsl</td>
<td>downgrade</td>
<td>good</td>
</tr>
<tr>
<td>2013-06-to-2012-06.xsl</td>
<td>downgrade</td>
<td>good</td>
</tr>
<tr>
<td>2015-01-to-2013-06.xsl</td>
<td>downgrade</td>
<td>good</td>
</tr>
</tbody>
</table>

### 17.6.1 Quality of transformations

#### Targets

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>/2003-FC/</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>/2007-06/</td>
<td>poor</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>/2008-02/</td>
<td>poor</td>
<td>poor</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>/2008-09/</td>
<td>poor</td>
<td>poor</td>
<td>poor</td>
<td>poor</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>/2009-09/</td>
<td>poor</td>
<td>poor</td>
<td>poor</td>
<td>poor</td>
<td>poor</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>/2010-04/</td>
<td>poor</td>
<td>poor</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>good</td>
<td>—</td>
</tr>
<tr>
<td>/2010-06/</td>
<td>poor</td>
<td>poor</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>good</td>
<td>—</td>
</tr>
<tr>
<td>/2011-06/</td>
<td>poor</td>
<td>poor</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>good</td>
<td>—</td>
</tr>
<tr>
<td>/2012-06/</td>
<td>poor</td>
<td>poor</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>good</td>
<td>—</td>
</tr>
<tr>
<td>/2013-06/</td>
<td>poor</td>
<td>poor</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>good</td>
<td>—</td>
</tr>
<tr>
<td>/2015-01/</td>
<td>poor</td>
<td>poor</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>fair</td>
<td>good</td>
<td>—</td>
</tr>
</tbody>
</table>

### 17.6.2 Key to quality

- **poor** (very lossy) - the bare minimum of metadata is preserved to allow image display, all other metadata is lost
- **fair** (lossy) - a portion of the metadata is preserved, at least enough to display the image and some other data, it will be far from complete however
- **good** - most information is preserved, it may be possible to do a better job but could be difficult for technical reasons or require custom code not just a transform
### 17.6.3 Matrix of transformation paths

This shows the sequence of transformations used to convert one version of the schema to another version.

<table>
<thead>
<tr>
<th>Source</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003-FC</td>
<td>—</td>
</tr>
<tr>
<td>2003-FC-TSF</td>
<td>code</td>
</tr>
<tr>
<td>2007-06</td>
<td>xslt: very lossy</td>
</tr>
<tr>
<td>2007-06-V2</td>
<td>xslt: very lossy</td>
</tr>
<tr>
<td>2008-02</td>
<td>xslt: very lossy</td>
</tr>
<tr>
<td>2008-02-V2</td>
<td>xslt: very lossy</td>
</tr>
<tr>
<td>2008-09</td>
<td>xslt: very lossy</td>
</tr>
<tr>
<td>2009-09</td>
<td>xslt: very lossy</td>
</tr>
<tr>
<td>2010-04</td>
<td>xslt: very lossy</td>
</tr>
<tr>
<td>2010-04-V2</td>
<td>xslt: very lossy</td>
</tr>
<tr>
<td>2010-06</td>
<td>soft: very lossy</td>
</tr>
<tr>
<td>2010-06-V2</td>
<td>soft: very lossy</td>
</tr>
<tr>
<td>2011-06</td>
<td>soft: very lossy</td>
</tr>
<tr>
<td>2012-06</td>
<td>soft: very lossy</td>
</tr>
<tr>
<td>2013-06</td>
<td>soft: very lossy</td>
</tr>
<tr>
<td>2014-01</td>
<td>soft: very lossy</td>
</tr>
</tbody>
</table>
Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

18.1 OMERO search

Beginning with 3.0-Beta3, the OMERO server will use Lucene\(^1\) to index all string and timestamp information in the database, as well as all OriginalFiles which can be parsed to simple text (see File parsers for more information). The index is stored under /OMERO/FullText (or the FullText subdirectory of your omero.data.dir, and can be searched with Google-like queries.

Once an entity is indexed, it is possible to start writing querying against the server via IQuery.findAllByFullText(). Use new Parameters(new Filter().owner()) and .group() to restrict your search. Or alternatively use the oma.api.Search interface (below).

See also:

Search and indexing configuration  
Section of the sysadmin documentation describing the configuration of the search and indexing for the server.

18.1.1 Field names

Each row in the database becomes a single Lucene Document parsed into the several Fields. A field is referenced by prefixing a search term with the field name followed by a colon. For example, name:myImage searches for myImage anywhere in the name field.

\(^1\)http://lucene.apache.org
<table>
<thead>
<tr>
<th>Field</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;field name&gt;</td>
<td>Any unprefixed field searches the combination of all fields together i.e. a search for <code>cell AND name:myImage</code> gets translated to <code>combined_fields:cell AND name:myImage</code>. Each string, timestamp, or Details field of the entity also gets its own Field entry, like the name field above.</td>
</tr>
<tr>
<td>details.owner.omeName</td>
<td>Login name of the owner of the object</td>
</tr>
<tr>
<td>details.owner.firstName</td>
<td>First name of the owner of the object</td>
</tr>
<tr>
<td>details.owner.lastName</td>
<td>Last name of the owner of the object</td>
</tr>
<tr>
<td>details.group.name</td>
<td>Group name of the owning group of the object</td>
</tr>
<tr>
<td>details.creationEvent.id</td>
<td>Id of the Event of this objects creation</td>
</tr>
<tr>
<td>details.creationEvent.time</td>
<td>When that Event took place</td>
</tr>
<tr>
<td>details.updateEvent.id</td>
<td>Id of the Event of this objects last modification</td>
</tr>
<tr>
<td>details.updateEvent.time</td>
<td>When that Event took place</td>
</tr>
<tr>
<td>details.permissions</td>
<td>Permissions in the form <code>rwrwrw</code> or <code>rw-</code></td>
</tr>
<tr>
<td>tag</td>
<td>Contents from a TagAnnotation.</td>
</tr>
<tr>
<td>annotation</td>
<td>Contents from any annotations, including TagAnnotation and any TextAnnotation on another TextAnnotation (a.k.a. a description). Non-string annotations like file annotations are not covered by this definition and are handled separately. See below.</td>
</tr>
<tr>
<td>annotation.ns</td>
<td>Namespace (if present) for any annotations on an object</td>
</tr>
<tr>
<td>annotation.type</td>
<td>Short type name, e.g. TextAnnotation or FileAnnotation for any annotations on an object</td>
</tr>
<tr>
<td>file.name</td>
<td>For FileAnnotations and objects they are attached to, the name of the OriginalFile</td>
</tr>
<tr>
<td>file.format</td>
<td>For FileAnnotations and objects they are attached to, the format of the OriginalFile</td>
</tr>
<tr>
<td>file.path</td>
<td>For FileAnnotations and objects they are attached to, the path of the OriginalFile</td>
</tr>
<tr>
<td>file.sha1</td>
<td>For FileAnnotations and objects they are attached to, the sha1 of the OriginalFile</td>
</tr>
<tr>
<td>file.contents</td>
<td>For FileAnnotations and objects they are attached to as well as the OriginalFile itself, the file contents themselves if their Format is configured with the File parsers.</td>
</tr>
<tr>
<td>${[NAME]}</td>
<td>For MapAnnotations and objects they are attached to, dynamic fields are generated for each of the NamedValue entries in the annotation. For example, if NamedValue(`temperature’, ‘37’) is one such value, a field named temperature will exist.</td>
</tr>
<tr>
<td>has_key</td>
<td>As ${NAME}, but a single field of name has_key is generated for each NamedValue entry with a value of the key such that a search for has_key:temperature in the example above is possible.</td>
</tr>
<tr>
<td>Internal</td>
<td>The default field prefix. Used by Hibernate Search to record the entity type. The class value, e.g. ome.model.core.Image is also entered in combined_fields. Unimportant for the casual users.</td>
</tr>
<tr>
<td>combined_fields</td>
<td>The default field prefix.</td>
</tr>
<tr>
<td>_hibernate_class</td>
<td>Used by Hibernate Search to record the entity type. The class value, e.g. ome.model.core.Image is also entered in combined_fields. Unimportant for the casual users.</td>
</tr>
<tr>
<td>id</td>
<td>The primary key of the entity. Unimportant for the casual user</td>
</tr>
</tbody>
</table>

### 18.1.2 Queries

Search queries are very similar to Google searches. When search terms are entered without a prefix (“name:”), then the default field will be used which combines all available fields. Otherwise, a prefix can be added to restrict the search.

### 18.1.3 Indexing

Successful searching depends on understanding how the text is indexed. The default analyzer used is the FullTextAnalyzer\(^2\).

---

\(^2\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/services/fulltext/FullTextAnalyzer.java
1. Desktop/image_GFP-H2B_1.dv ---> "desktop", "image", "gfp", "h2b", "1", "dv"
3. Desktop/image_GFP_01-H2B.dv ---> "desktop", "image", "gfp", "01", "h2b", "dv"

Assuming these entries above for Image.name:

- searching for GFP-H2B returns 1 and 2.
- searching for “GFP H2B” also returns 1 and 2.
- searching for GFP H2B returns 1, 2, and 3, since the two terms are joined by an OR.

18.1.4 Information for developers

ome.api.IQuery

The current IQuery implementation restricts searches to a single class at a time.

- findAllByFullText(Image.class, "metaphase") – Images which contain or are annotated with “metaphase”
- findAllByFullText(Image.class, "annotation:metaphase") – Images which are annotated with “metaphase”
- findAllByFullText(Image.class, "tag:metaphase") – Images which are tagged with “metaphase” (specialization of the previous)
- findAllByFullText(Image.class, "file.contents:metaphase") – Images which have files attached containing “metaphase”
- findAllByFullText(OriginalFile.class, "file.contents:metaphase") – File containing “metaphase”

ome.api.Search

The Search API offers a number of different queries along with various filters and settings which are all maintained on the server.

The matrix below show which combinations of parameters and queries are supported (S), will throw an exception (X), and which will simply silently be ignored (I).

<table>
<thead>
<tr>
<th>Query Method →</th>
<th>byGroupForT-agS/byT-agSForGroup</th>
<th>byFullText/Some</th>
<th>byAnnotatedWithMustNone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>annotated between</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>annotated by</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>created between</td>
<td>S</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>modified between</td>
<td>S</td>
<td>I (Im-mutable)</td>
<td>S</td>
</tr>
<tr>
<td>owned by</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>all types</td>
<td>X</td>
<td>I</td>
<td>X</td>
</tr>
<tr>
<td>1 type</td>
<td>S</td>
<td>I</td>
<td>S</td>
</tr>
<tr>
<td>N types</td>
<td>X</td>
<td>I</td>
<td>X</td>
</tr>
<tr>
<td>only ids</td>
<td>S</td>
<td>I</td>
<td>S</td>
</tr>
<tr>
<td>Ordering / Fetches</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>orderBy</td>
<td>S</td>
<td>I</td>
<td>S</td>
</tr>
<tr>
<td>fetchAnnotations</td>
<td>3</td>
<td>I</td>
<td>4</td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>setProjections 5</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>current*Metadata 6</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

3 Any fetchAnnotation() argument to byFullText() or related queries, returns all annotations.
4 byAnnotatedWith() does not accept a fetchAnnotation() argument of Annotation.class.
Leading wildcard searches

Leading wildcard searches are disallowed by default. “?omething” or “*hatever”, for example, would both throw exceptions. They can be run by using:

```java
Search search = serviceFactory.createSearchService();
search.setAllowLeadingWildcards(true);
```

There is a performance penalty, however. In addition, wildcard searches get expanded on the server to boolean queries. For example, assuming “ACELL”, “BCELL”, and “CCELL” are all terms in your index, then the query:

```
*CCELL
```

gets expanded to:

```
ACELL OR BCELL OR CCELL
```

If there are more than `omero.search.maxclause` terms in the expansion (default is 4096), then an exception will be thrown. This requires the user to enter a more refined search, but not because there are too many results, only because there is not enough room in memory to search on all terms at once.

Extension points

Two extension points are currently available for searching. The first are the File parsers mentioned above. By configuring the map of Formats (roughly mime-types) of files to parser instances, extracting information from attached binary files can be made quick and straightforward.

Similarly, Search bridges provide a mechanism for parsing all metadata entering the system. One built in bridge (the Full-TextBridge) parses out the fields mentioned above, but by creating your own bridge it is possible to extract more information specific to your site.

See also:
Structured annotations, Search bridges, File parsers, Query Parser Syntax.

Luke, a Java application which you can download and point at your /OMERO/FullText directory to get a better feeling for Lucene queries.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

18.2 File parsers

File parsers extract text from various file types and provide it as a Reader to the FullTextBridge for use during search indexing. Plain text formats can use the default fileParser bean, but any specialized format, such as PDF or RTF requires special libraries and special registration.

18.2.1 Configuration

Currently, configuration takes places solely in service-ome.api.Search.xml. Eventually, it should be able to replace file parsers at configuration or even runtime.

---

5 setProjects may need to be removed if Lucene cannot handle OMERO’s security requirements.
6 Not yet implemented.
7 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/services/fulltext/FullTextBridge.java
8 http://lucene.apache.org/core/3_6_0/queryparsersyntax.html
9 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/resources/ome/services/service-ome.api.Search.xml

---
18.2.2 Available parsers

<table>
<thead>
<tr>
<th>File type</th>
<th>Parser</th>
</tr>
</thead>
<tbody>
<tr>
<td>application/pdf</td>
<td><a href="http://pdfbox.apache.org">http://pdfbox.apache.org</a></td>
</tr>
<tr>
<td>text/xml</td>
<td>(internal)</td>
</tr>
<tr>
<td>text/plain</td>
<td>(internal)</td>
</tr>
<tr>
<td>text/csv</td>
<td>(internal)</td>
</tr>
</tbody>
</table>

The base class for File parsers are FileParser.java\textsuperscript{11}

See also: OME\textsuperscript{RO} search

Note: This documentation is for OME\textsuperscript{RO} 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OME\textsuperscript{RO} 5.3 is expected before the end of 2016.

18.3 Search bridges

A “bridge” is the mapping between your metadata and how it is stored in the Lucene\textsuperscript{12} index. OME\textsuperscript{RO} search uses one internal bridge to parse all of your metadata for later searching. If, however, there is more metadata that you would like to add to the index, you can implement the org.hibernate.search.bridge.FieldBridge interface yourself, or subclass the helper class components/server/src/ome/services/fulltext/BridgeHelper.java\textsuperscript{13}

18.3.1 Example

Assume you wanted to be able to search for a project based on the name of all images contained in that project. In the set method,

```
public void set(final String name, final Object value,
                final Document document, final Field.Store store,
                final Field.Index index, final Float boost) {
```

you would need to add a field to the Document for each Image.

```
    Project p = (Project) value;
    List<Image> images = getImages(p);
    for (Image image : images) {
        add(document, "image_name", image.getName(), store, index, boost);
    }
```

18.3.2 Configuration

Custom bridges are configured in Search but can be overridden via the standard configuration mechanisms. The omero.search.bridges property defines a comma-separated list of bridge classes which will be passed to components/server/src/ome/services/fulltext/FullTextBridge.java\textsuperscript{14}

See Java deployment for how to have your bridge classes included on the server’s classpath if it doesn’t get built by the Build System.

\textsuperscript{11}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/services/fulltext/FileParser.java
\textsuperscript{12}http://lucene.apache.org
\textsuperscript{13}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/services/fulltext/BridgeHelper.java
\textsuperscript{14}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/services/fulltext/FullTextBridge.java
18.3.3 Available bridges

See components/server/src/ome/services/fulltext/bridges\(^\text{15}\) for a list of provided (example) bridges.

18.3.4 Re-indexing

BridgeHelper provides two methods – `reindex(IObject)` and `reindexAll(List<IObject>)` – for keeping the indexes for objects in sync.

For example, if the `image.name` above were to be changed, the index for the `Project` would be stale until the `Project` itself were re-indexed. Custom bridges can call `reindex(Project)` while indexing the image to have the `Project` re-indexed from the backlog. Before any new changes are processed, the backlog is always first cleared. One caveat: while processing the backlog, no new objects can be added to it.

For bridge writers, this means concretely that implementations should check for all related types and index them in groups, rather than relying on transitivity. For example,

```java
if (value instanceof Project) {
    final Project p = (Project) value;
    handleProject(p, document, store, index, boost);
    for (final ProjectDatasetLink pdl : p.unmodifiableDatasetLinks()) {
        final Dataset d = pdl.child();
        reindex.add(d);
        handleDataset(d, document, store, index, boost);
    }
}
else if (value instanceof Dataset) {
    final Dataset d = (Dataset) value;
    handleDataset(d, document, store, index, boost);
    for (final ProjectDatasetLink pdl : d.unmodifiableProjectLinks()) {
        final Project p = pdl.parent();
        reindex.add(p);
        handleProject(p, document, store, index, two_step_boost);
    }
}
else if (value instanceof Image) {
    final Image i = (Image) value;
    handleImage(document, store, index, two_step_boost, i);
    for (final DatasetImageLink dil : i.unmodifiableDatasetLinks()) {
        final Dataset d = dil.parent();
        reindex.add(d);
        handleDataset(d, document, store, index, boost);
        for (final ProjectDatasetLink pdl : d.unmodifiableProjectLinks()) {
            final Project p = pdl.parent();
            reindex.add(p);
        }
    }
}
```

\(^{15}\)https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/server/src/ome/services/fulltext/bridges
handleProject(p, document, store, index, boost);
}
}

// Handle re-indexing
if (reindex.size() > 0) {
    reindexAll(reindex);
}

In which case, regardless of whether an Image, Dataset, or Project is indexed, all related objects are simultaneously added to the backlog, which will be processed in the next cycle, but their indexing will not add any new values to the backlog.

See #955\(^{16}\) and #1102\(^{17}\)

See also:

*OMERO search*

\(^{16}\)https://trac.openmicroscopy.org/ome/ticket/955
\(^{17}\)https://trac.openmicroscopy.org/ome/ticket/1102
19.1 Password Provider

A Password Provider is an implementation of the Java interface `ome.security.auth.PasswordProvider`. Several implementations exist currently:

- `ome.security.auth.JdbcPasswordProvider` is the most common provider, and uses the “password” table for storing passwords hashed using MD5 and salt per user.
- `ome.security.auth.FilePasswordProvider` is rarely used, but in some scenarios may be useful since it permits setting usernames and passwords in a plain text file.
- `ome.security.auth.LdapPasswordProvider` is a highly configurable provider which provides READ-ONLY access to an LDAP server and can create users and groups on the fly. See `LDAP plugin design` for more information.

The “chainedPasswordProvider” (`ome.security.auth.PasswordProviders`) is configured for use by default in Security under `omero.security.password_provider`. It first checks with the LdapPasswordProvider and then falls back to the JdbcPasswordProvider.

To write your own provider, you can either subclass from `ome.security.auth.ConfigurablePasswordProvider` as the providers above do, or write your own implementation from scratch. You will need to define your object in a Spring XML file matching the pattern `omero/services/db-*.xml`. See `Extending OMERO.server` more for information.

19.1.1 Things to keep in mind

- All the existing implementations take care to publish a `LoginAttemptMessage` so that any `LoginAttemptListener` implementation can properly react to failed logins. Your implementation should probably do the same.
- When dealing with chains of password providers, an implementation can safely return null from `checkPassword` to say “I don’t know anything about this”. This is only important if you configure your own chained password provider with your new implementation as one of the elements.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

---

7. https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/services/messages/LoginAttemptMessage.java
19.2 LoginAttemptListener

All the Password Provider implementations provided by default publish a “LoginAttemptMessage” every time they check a password value. This permits any org.springframework.context.ApplicationListener<LoginAttemptMessage> to react to the login. Only one implementation is active by default (as of 4.2.1): ome.security.auth.LoginAttemptListener which throttles logins after a given number of failed attempts. Configuration for this listener is available in Security:

ome.security.login_failure_throttle_count=1 # Number of failed attempts before throttling begins
ome.security.login_failure_throttle_time=3000 # Time in milliseconds

A more sophisticated listener would lock the user’s account until an administrator intervenes. This is the goal of #3139.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

19.3 LDAP plugin design

Once configured, LDAP authentication allows sysadmins to control OMERO’s user and group creation via an external, locally-maintained LDAP server. Due to the flexibility of LDAP, each instance may have a number of requirements that cannot be supported out of the box. Below, we discuss the design of the LDAP plugin as well as how it can be extended for local use.

19.3.1 Simple walkthrough

The LDAP plugin follows these steps:

1. Sysadmin configures properties mapping users and groups from LDAP to OMERO.

2. Once LDAP is enabled, any OMERO user who has a value of true in the ldap column of the experimenter table will have their password checked against LDAP and not against OMERO (changing the password via OMERO is not supported). This functionality is provided by the Password Provider. The DN (Distinguished Name) is not stored in the OMERO DB and is retrieved from the LDAP server on each user login.

3. If there is no OMERO user for a given name, the LDAP plugin will use omero.ldap.user_filter and omero.ldap.user_mapping to look for a valid user:

   (a) The user_mapping property is of the form: omeName=<ldap attribute>;firstName=<ldapAttribute>;…

   (b) For looking up new users, the plugin will only use the omeName attribute. For example, if a user tries to login with “emma” and the user_mapping starts with omeName=cn; then the LDAP search will be for (cn=emma).

   (c) The (cn=emma) LDAP filter is then added to the value of omero.ldap.user_filter. For example, if the filter is (objectClass=inetOrgPerson), the full query for the new user will be: ((&objectClass=inetOrgPerson)(cn=emma))

4. If the search returns a single LDAP user, then an OMERO user will be created with all properties mapped according to omero.ldap.user_mapping and the ldap property set to true.

5. Then the user will be placed in groups according to the value of omero.ldap.new_user_group, which are created if necessary. Details of the various options can be found under LDAP authentication. Each option is handled by a different NewUserGroupBean implementation.

---

8https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/services/messages/LoginAttemptMessage.java
9https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/security/auth/LoginAttemptListener.java
10https://trac.openmicroscopy.org/ome/ticket/3139
19.3.2 NewUserGroupBean.java

The interface described for the "bean: new_user_group" prefix, is ome.security.auth.NewUserGroupBean\(^{11}\). It defines a single method: `groups(..., AttributeSet set)` which returns a list of ExperimenterGroup ids (List<Long>) which the user should be added to.

Other prefix handlers also implement the interface as examples. In the same package are:

- :attribute: - AttributeNewUserGroupBean\(^{12}\)
- :ou: - OrgUnitNewUserGroupBean\(^{13}\)
- :query: - QueryNewUserGroupBean\(^{14}\)

See also:

- **OMERO.server installation** Instructions for installing OMERO.server on UNIX and UNIX-like platforms
- **OMERO.server installation** Instructions for installing OMERO.server on Windows platforms
- **Server security and firewalls** General instructions on server security

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

19.4 OMERO roles

There are two areas where roles are used. The first is in service-level security (deciding who can make what calls) and the second is in object-level security (who can read and edit individual objects). Both of these sets of roles are composed of “ExperimenterGroups”.

19.4.1 Setting roles

An Experimenter is given a role by being a member of an ExperimenterGroup (specifically, this means that there exists a GroupExperimenterMap where child == the experimenter id and parent == the experimenter group id). Creating a GroupExperimenterMap is generally done transparently by IAdmin service. Instead, administrators call:

- IAdmin.createUser(user)
- IAdmin.createGroup(group)
- IAdmin.addGroups(user, group, group, …)
- IAdmin.removeGroups(user, group, group, …)
- IAdmin.createSystemUser(user)

19.4.2 Service-level

The two main roles that are distinguished at the service-level are “system” and “user” groups. These groups are created during installation and must not be configured by administrators. All users added through IAdmin.createUser(user) are automatically added to the “user” group, and all users added through IAdmin.createSystemUser(user) are added to both “system” and “user” groups.

During login, a user is checked against all groups for membership in “user” or “system”, and no special action needs to be taken by the user or client developer.

**Note:** Although currently all methods in the session beans are labelled as @RolesAllowed("user") or @RolesAllowed("system"), there is nothing stopping a developer from writing a service method which accepts another role, as long as that role has been created in the ExperimenterGroup table.

\(^{11}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/security/auth/NewUserGroupBean.java
\(^{12}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/security/auth/AttributeNewUserGroupBean.java
\(^{13}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/security/auth/OrgUnitNewUserGroupBean.java
19.4.3 Object-level

Object-level security is more complicated. When execution reaches the EventHandler, a second login takes place to authorize the user with the OMERO security system. This second authorization process takes into account the group that (can be) passed into the client ServiceFactory\ (Login) via Login(String, String, String, String). If a user has not set the group name or the default “user” group has been set, then the default group for that user will be used (a user is not allowed to use the “user” group for object updates). If the group is set to “system”, then the “system” group will be used, and a user is granted admin privileges for object updates. This means that a user could be authorized to call a method by being in the “system” group, but if the “system” group is not specified, SecurityViolations will most likely be thrown.

19.4.4 Special privileges for PIs

There is one other special, implicit role which is group leader. The user listed as “owner” for a group is considered the group leader, also known as the PI (principal investigator) of that group. For all objects that are assigned to that group, the PI has near-admin access. Objects which are set to unreadable (“-wu-wu-wu”) will still be visible to the PI. The same objects can also be updated regardless of the permissions set.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

19.5 OMERO security system

The OMERO security system is intended to be as transparent as possible while permitting users to configure the visibility of their data. For the user, this means that with no special actions, data and metadata created will be readable by both members of the same group and by other users, but will be writable by no one, comparable to a umask of 755 on Unix. For the developer, transparency means that there is little or no code that must be written to prevent security violations, and simple mechanisms for allowing restricted operations when the time comes.

Other links which may be of use:

- OMERO admin interface
- OMERO roles
- Groups and permissions system
- OMERO permissions history, querying and usage

19.5.1 Concepts

Several concepts and/or components from our and other code bases play a role in OMERO security:

**Hibernate Listeners and Events** listeners and events are the two extension points provided by Hibernate for responding to and influencing internal actions. Essentially any method on the org.hibernate.Session interface has a corresponding event, and almost the same is true for the interceptor. Additionally interceptors can change the state of the objects before INSERT and UPDATE, and after SELECT.

**Hibernate Filters** filters are a mechanism for injecting SQL clauses into the SELECT statements generated by Hibernate. Similar to listeners and events for write actions, filters allow us to extend Hibernate functionality with our own logic.

**Handler/interceptor** as outlined in Aspect-oriented programming, OMERO makes extensive use of method interceptors to relieve the developer of some coding burden. Transactions, session management, and, naturally, security are handled largely by our interceptors (or “handlers”).

**Events** Every write action produces an Event in the database. This database contains several EventLogs which specify exactly what was created or altered during that specific event.
19.5.2 Participants

Now, with the concepts cleared up, we can take a look at all of the concrete source artifacts ("participants") which are important for security.

Top-level and build

omeo.properties\textsuperscript{17} contains login and connection information for the database and OMERO.

build.properties.example\textsuperscript{18} contains the default root password. This can be overridden with your own etc/local.properties file.

hibernate.properties\textsuperscript{19} contains default connection information for the database, this includes the user name and if necessary the user password. These values can be overridden in local.properties.

omeo.properties\textsuperscript{20} contains a default user group, event type, and connection information for logging in from the client side, if no Login or Server is specified to ServiceFactory. These values can be overridden in local.properties.

mapping.vm\textsuperscript{21} specifies the default permissions that all objects will have after construction, as well as attaches the security filter to all classes and collections.

data.vm\textsuperscript{22} used by DSLTask to generate psql-footer.sql which is used to bootstrap the database security system (root et al).

common/build.xml\textsuperscript{23} contains an ant target (adduser) which will create a user and empty password from the command line. This target can also be called from the top-level (java omero adduser).

Client and common

the server uses the information in /etc/local.properties to create a Login object. If no Login, Server, or Properties is provided to the ServiceFactory constructor, the empty properties defined in ome/config.xml\textsuperscript{24} is used.

IAdmin.java\textsuperscript{25} main interface for administering accounts and privileges. See OMERO admin interface for more.

ITypes.java\textsuperscript{26} only related to security by necessity. The security system disallows the creation of certain “System-Types”. Enumerations are one of these. ITypes, however, provides a createEnumeration method with general access.

GraphHolder.java\textsuperscript{27} all model objects (implementations of IObject have a never-null GraphHolder instance available. This graphholder is responsible for various OMERO and Hibernate internal processes. One of these is the exchange of Tokens. For the server, the existence of a special token within the GraphHolder grants certain privileges to that IObject. This logic is encapsulated within the SecuritySystem.

Details.java\textsuperscript{28} contains all the fields necessary to perform access control, such as owner, group, and permissions.

Permissions.java\textsuperscript{29} representation of rights and roles. For more information, see Groups and permissions system.

Token.java\textsuperscript{30} an extremely simple class ("public class Token []") which is only significant when it is equivalent ("==") to a privileged Token stored within the SecuritySystem.

IEnum.java\textsuperscript{31} the only non-access control related types which are considered “System-Types” are enumerations. IEnum is a marker interface for all enumerations and creation of IEnum implementations can only be performed through ITypes.

SecurityViolation.java\textsuperscript{32} the exception thrown by the OMERO security system at the first hint of misdoings.

Principal.java\textsuperscript{33} an Omero-specific implementation of the java.security.Principal interface. Carries in addition to the typical name field, information about the user group, the event type, and the session umasks.

meta.ome.xml\textsuperscript{34}

JBoss-only

ServiceFactory.java\textsuperscript{35} Login.java\textsuperscript{36} Server.java\textsuperscript{37}

\textsuperscript{24}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/resources/ome/config.xml
\textsuperscript{25}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/resources/mappings/meta.ome.xml
\textsuperscript{26}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/system/ITypes.java
\textsuperscript{27}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/system/GraphHolder.java
\textsuperscript{28}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/system/Details.java
\textsuperscript{29}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/system/Permissions.java
\textsuperscript{30}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/system/Token.java
\textsuperscript{31}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/system/IEnum.java
\textsuperscript{32}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/system/SecurityViolation.java
\textsuperscript{33}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/system/Principal.java
\textsuperscript{34}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/system/Meta.java
\textsuperscript{35}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/system/ServiceFactory.java
\textsuperscript{36}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/system/Login.java
\textsuperscript{37}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/system/Server.java
19.5.3 End-to-end

Build system

Security starts with the build system and installation. During the generation of the model (by the DSLTask), a sql script is created called “data.sql”. After ddl.sql creates the database, data.sql bootstraps the security system by creating the initial (root) experimenter, and event, and then creates the “system” group and the “user” group. It then creates a password table and sets the root password to “ome”. (It also creates all of the enumeration values, but that is unimportant for security).

Note: The password table is not mapped into Hibernate, and is only accessible via the OMERO admin interface.

Client-side

To begin the runtime security process, a user logs in by providing a Login and/or a Server instance to ServiceFactory. These types are immutable and their values remain constant for the lifetime of the ServiceFactory. The user can also set the umask property on ServiceFactory_. This value is mutable and can be set at anytime.

The values are converted to java.util.Properties which are merged with the properties from the *.properties files from /etc to create the client OmeroContext (also known as the “application context”). The context contains a Principal and user credentials (password etc.) which are associated with the thread before each method execution in a specialized TargetSource. Finally, these objects are serialized to the application server along with the method arguments.

Application server

The application server first performs one query (most likely SQL) to check that the credentials match those for the given user name. A second query is executed to retrieve all roles/groups for the given user. If the roles returned are allowed to invoke the desired method, invocation continues with the queried user and roles stored in the InvocationContext.

Server code

Execution then passes to OMERO code, specifically to the interceptors and lifecycle methods defined on our session beans. This intercepting code checks the passed Principal for OMERO-specific information. If this information is available, it is passed into the SecuritySystem through the login method. Finally, execution is returned to the actual bean which can either delegate to OMERO services or perform logic themselves.
Interceptors

All calls to the delegates (and in the future all calls on the session beans) are also caught intercepted by Spring-configured interceptors. These guarantee that the system is always in a valid and secure state. In stack order they are:

- the service handler, which handles logging and checks all arguments against ServiceInterface annotations;
- the proxy handler, which after execution, removes all uninitialized Hibernate objects to prevent exceptions (special logic allows this to happen for unloaded objects);
- the transaction handler, which binds a transaction to the thread,
- the session handler, which uses the now prepared transaction to initialize either a new or a cached (in the case of stateful session beans) session and also bind it to the thread;
- and finally, the event handler, which performs what one might actually consider login. It instantiates Experimenter, ExperimenterGroup, and Event objects from Hibernate and gives them a special Token so that they can authenticate themselves later to the SecuritySystem and turns session read security on for the entirety of execution below its frame.

Services

Finally execution has reached the OMERO services and can begin to perform logic. Because of these layers, almost no special logic (other than eviction and not calling write methods from within read methods. see #22355) needs to be considered. There are, however, a few special cases.

IQuery (within the application server), for example will always return a graph of active Hibernate objects. Changes to them will be persisted to the database on flush.

IUpdate, on the other hand, does contain some logic for easing persistence, though this will eventually be ported to the Hibernate event system. This includes pre-saving the newly created event and the work of UpdateFilter like reloading objects unloaded by the proxy handler (above).

Finally, IAdmin is special in that it and it alone access the non-Hibernate password data store and even access application server APIs (like JMX) in order to make authentication and authorization function properly.

Hibernate

Once execution has left this service layer, it enters the world of Hibernate ORM. Here we cannot actively change functionality but only provide callbacks like the OmeroInterceptor and EventListeners. The OmeroInterceptor instance registered with the HibernateSessionFactory (via Spring) is allowed for calling back to the often mentioned SecuritySystem to determine what objects can be saved and which deleted. It also properly sets the, for a user mostly unimportant, Details object. The EventListeners are more comprehensive than the OmeroInterceptor and can influence almost every phase of the Hibernate lifecycle, specifically every method on the Session interface.

The event listeners which implement AbstractSaveEventListener (i.e. MergeEventListener, SaveOrUpdateEventListener, etc.) are responsible for reloading unloaded objects (and will hopefully take this functionality fully from IUpdate) and provide special handling for enums and other system types. There are also event listeners which are the equivalent of database triggers (pre-update, post-delete, etc.) and these are used for generating our audit log.

So much for write activities. Select queries are, as mentioned above, secured through the use of Hibernate filters which add join and where clauses dynamically to queries. For example an HQL query of the form:

```
select i from Image i
```

would be filtered so that the current user does not receive references to any objects with reduced visibility:

```
select i from Image i where ( current_user = :root OR i.permissions = :readable )
```

The actual clauses added are much more complex and are added for each joined entity type (i.e. table) which appears in a query.

55https://trac.openmicroscopy.org/ome/ticket/223
select i from Image i join i.defaultPixels p

would contain the "( current_user = :root ... )" clause twice.

Currently, subqueries are an issue in that the clauses do not get added to them. This may cause consternation for some particular queries.

Security system

All of this is supported by an implementation of the SecuritySystem interface which encapsulates all logic regarding security. It also hides as much as it can, and if not specifically needed should be ignored. However, before you attempt to manually check security, by all means use the security system, and for that, it may need to be acquired from the server-side OmeroContext. Currently, there is no client-side security system. See #234.

The OMERO security system and its current only implementation BasicSecuritySystem? are somewhat inert and expect well-defined and trusted (see #235) methods to invoke callbacks during the proper Hibernate phase.

19.5.4 Logging in (client-side)

When using the client library and the ServiceFactory, logging in is trivial. One need only set several System properties or place them in an omero.properties file somewhere on the classpath. Internally, Spring takes the System properties and creates an ome.system.Principal instance. This is then passed to the server on each invocation of a proxy obtained from JNDI.

19.5.5 Logging in (server-side)

Much of this infrastructure is not available to server-side code (no ome/client/spring.xml, no ServiceFactory, etc.). As such, the Principal needs to be manually created and provided to the server-side SecuritySystem.java.

Basically it amounts to this:

Principal p = new Principal( omeroUserName, omeroGroupName, omeroEventTypeValue );
securitySystem.login( p );

This must be run otherwise the EventHandler will throw a security exception.

Note: The code above is being run in a secure context (i.e. you are root). Please be careful.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

19.6 OMERO permissions history, querying and usage

19.6.1 Introduction

The OMERO permissions model has had a significant overhaul from version 4.1.x to 4.4.x. Users and groups have existed in OMERO since well before the initial 4.1.x releases and numerous permissions levels were possible in the 4.1.x series but it was largely assumed that an Experimenter belonged to a single Group and that the permissions of that Group were private.

The OMERO permissions system received its first significant update in 4.2.0 with the introduction of multiple group support throughout the platform and group permissions levels.

---

56https://trac.openmicroscopy.org/ome/ticket/234
57https://trac.openmicroscopy.org/ome/ticket/235
58https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/system/Principal.java
60https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/security/basic/EventHandler.java
In a 4.1.x object graph Group containment was not enforced i.e. two linked objects (such as a Project and Dataset) could in theory be members of two distinct Groups. All objects continued to carry their permissions and those permissions were persisted in the database.

**Things to note about 4.2.x permissions**

- Objects could not be moved between groups easily.
- It was not possible to reduce the permissions level of a group.
- The delete service (introduced in OMERO 4.2.1) was made aware of the permissions system.
- ‘Default Group’ switching was required to make queries in different permissions contexts.

**Note:** Queries span only one group at a time. Inserts and updates as other users must be done by creating a session as that user.

**See also:**
OMERO 4.2.0 Server Permissions\(^61\) Database upgrade from 4.1 to 4.2\(^62\) Deleting in OMERO

**Changes for OMERO 4.4.x**

The second major OMERO permissions system innovations were performed in 4.4.0:

- Cross group querying was reintroduced.
- Change group was enabled, allowing the movement of graphs of objects between groups.
- Permissions level reduction was made possible for read-annotate to read-only transitions.
- A thorough user interface review resulted in the following features being made available in the UI:
  - single group browsing and user-switching (available since 4.4.0)
  - browsing data across multiple groups (available since 4.4.6 and refined in 4.4.7)
- The concept of a ‘Default or Primary Group’ was deprecated.

**Note:** Queries, inserts and updates span any or all groups and any user via options flags.

---

### 19.6.2 Working with the OMERO 5.2.6 permissions system

**Example environment**

- OMERO 5.2.6 server
- IPython shell initiated by running `omero shell --login`

**Group membership**

<table>
<thead>
<tr>
<th>User</th>
<th>private-1</th>
<th>read-only-1</th>
<th>read-write-1</th>
<th>read-annotate-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>user-2</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>user-3</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

---

\(^61\) [http://www.openmicroscopy.org/site/support/previous/omero420/server/permissions](http://www.openmicroscopy.org/site/support/previous/omero420/server/permissions)

\(^62\) [http://www.openmicroscopy.org/site/support/previous/omero420/server/db-upgrade-41-to-42](http://www.openmicroscopy.org/site/support/previous/omero420/server/db-upgrade-41-to-42)
Simple inserts and queries

While the ‘Default Group’ is essentially a deprecated concept, a user must be logged into one to provide a default context. It is still possible to change this default group but it is no longer required to make queries in other permissions contexts.

All remote calls to an OMERO server, since well before version 4.1.x, have the option of taking an Ice context object. Through this object, and manipulations thereof, we can affect our query context. What follows is a series of examples exploring inserts and queries using contexts that span a single group at a time.

Retrieving a user’s event context and group membership

```python
#!/python
# Session that has already been created for user-2
session = client.getSession()

# Retrieve the services we are going to use
admin_service = session.getAdminService()

ec = admin_service.getEventContext()
print(ec)
groups = [admin_service.getGroup(v) for v in ec.memberOfGroups]
for group in groups:
    print('Group name: %s' % group.name.val)
```

Example output:

```
object #0 (::omero::sys::EventContext)
{
    shareId = -1
    sessionId = 1783
    sessionUuid = 213adc46-2c5f-449b-81fc-fe24dec38b58
    userId = 10
    userName = user-2
    groupId = 9
    groupName = private-1
    isAdmin = False
    eventId = -1
    eventType = User
    memberOfGroups =
    {
        [0] = 9
        [1] = 8
        [2] = 1
    }
    leaderOfGroups =
    {
    }
    groupPermissions = object #1 (::omero::model::Permissions)
    {
        _restrictions =
        {
        }
        _perm1 = -120
    }
}
Group name: private-1
Group name: read-only-1
Group name: user
```
Here you can see and validate that, when logged in as user-2, we are a member of both the private-1 and read-only-1 groups. Membership of the user group is required in order to login. This group essentially acts as a role, letting the OMERO security system know whether or not the user is active.

Inserting and querying data from specific groups

For the purposes of this example, we will prepare a single Project in both the private-1 and read-only-1 groups and then perform various queries on those Projects.

```python
# Session that has already been created for user-2
session = client.getSession()

# Project object instantiation
private_project = ProjectI()
private_project.name = rstring('private-1 project')
read_only_project = ProjectI()
read_only_project.name = rstring('read-only-1 project')

# Retrieve the services we are going to use
update_service = session.getUpdateService()
admin_service = session.getAdminService()
query_service = session.getQueryService()

# Groups we are going to write data into
private_group = admin_service.lookupGroup('private-1')
read_only_group = admin_service.lookupGroup('read-only-1')

# Save and return our two projects, setting the context correctly for each
ctx = {'omero.group': str(private_group.id.val)}
private_project = update_service.saveAndReturnObject(private_project, ctx)
ctx = {'omero.group': str(read_only_group.id.val)}
read_only_project = update_service.saveAndReturnObject(read_only_project, ctx)

private_project_id = private_project.id.val
read_only_project_id = read_only_project.id.val

print 'Created Project:%d in group private-1' % (private_project_id)
print 'Created Project:%d in group read-only-1' % (read_only_project_id)

# Query for the private project we created using private-1
# You will notice that this returns the Project as we have specified
# the group that the Project is in within the context passed to the
# query service.
ctx = {'omero.group': str(private_group.id.val)}
params = ParametersI()
params.addId(private_project_id)
projects = query_service.findAllByQuery(
    'select p from Project as p ' \
    'where p.id = :id', params, ctx)

print 'Found %d Project(s) with ID %d in group private-1' % (len(projects), private_project_id)

# Query for the private project we created using read-only-1
# You will notice that this does not return the Project as we have **NOT**
# specified the group that the Project is in within the context
# passed to the query service.
ctx = {'omero.group': str(read_only_group.id.val)}
params = ParametersI()
params.addId(private_project_id)
projects = query_service.findAllByQuery(
    'select p from Project as p ' \
    'where p.id = :id', params, ctx)

print 'Found %d Project(s) with ID %d in group read-only-1' % \
    (len(projects), private_project_id)

# Use the OMERO 4.3.x introduced delete service to clean up the Projects
# we have just created.
handle = session.submit(Delete('/Project', private_project_id, None))
try:
    callback = CmdCallbackI(client, handle)
    callback.loop(10, 1000)  # Loop a maximum of ten times each 1000ms
finally:
    # Safely ensure that the Handle to the delete request is cleaned up,
    # otherwise there is the possibility of resource leaks server side that
    # will only be cleaned up periodically.
    handle.close()

handle = session.submit(Delete('/Project', read_only_project_id, None))
try:
    callback = CmdCallbackI(client, handle)
    callback.loop(10, 1000)  # Loop a maximum of ten times each 1000ms
finally:
    handle.close()

Example output:

Created Project:113 in group private-1
Created Project:114 in group read-only-1
Found 1 Project(s) with ID 113 in group private-1
Found 0 Project(s) with ID 113 in group read-only-1

Advanced queries

In OMERO 4.4.0, cross group querying was reintroduced. Again, we make use of the Ice context object. Through this object, and manipulations thereof, we can expand our query context to span all groups via the use of -1. What follows is a series of example queries using contexts that span all groups.

Querying data across groups

#!/python
from omero.model import *
from omero.rtypes import *
from omero.sys import ParametersI
from omero.cmd import Delete, DoAll
from omero.callbacks import CmdCallbackI

# Session that has already been created for user-2
session = client.getSession()

# Project object instantiation
private_project = ProjectI()
private_project.name = rstring('private-1 project')
read_only_project = ProjectI()
read_only_project.name = rstring('read-only-1 project')

# Retrieve the services we are going to use
update_service = session.getUpdateService()
admin_service = session.getAdminService()
query_service = session.getQueryService()

# Groups we are going to write data into
private_group = admin_service.lookupGroup('private-1')
read_only_group = admin_service.lookupGroup('read-only-1')

# Save and return our two projects, setting the context correctly for each.
# ALL interactions with the update service where NEW objects are concerned
# must be passed an explicit context and NOT '-1'. Otherwise the server
# has no idea which set of owners to assign to the object when persisted.
ctx = {'omero.group': str(private_group.id.val)}
private_project = update_service.saveAndReturnObject(private_project, ctx)
ctx = {'omero.group': str(read_only_group.id.val)}
read_only_project = update_service.saveAndReturnObject(read_only_project, ctx)

private_project_id = private_project.id.val
read_only_project_id = read_only_project.id.val
print 'Created Project:%d in group private-1' % (private_project_id)
print 'Created Project:%d in group read-only-1' % (read_only_project_id)

# Query for the private project we created using private-1
# You will notice that this returns both Projects as we have specified
# '-1' in the context passed to the query service.
ctx = {'omero.group': '-1'}
params = ParametersI()
params.addIds([private_project_id, read_only_project_id])
projects = query_service.findAllByQuery(
    'select p from Project as p \n    where p.id in (:ids)', params, ctx)

print 'Found %d Project(s)' % (len(projects))

Use the OMERO 4.3.x introduced delete service to clean up the Projects
we have just created. The delete service uses '-1' by default for all its
internal queries. We are also introducing the 'DoAll' command, which
allows for the aggregation of 'Delete' commands.
delete_requests = [
    Delete('Project', private_project_id, None),
    Delete('Project', read_only_project_id, None)
]
handle = session.submit(DoAll(delete_requests))
try:
    callback = CmdCallbackI(client, handle)
callback.loop(10, 1000)  # Loop a maximum of ten times each 1000ms
finally:
    # Safely ensure that the Handle to the delete request is cleaned up,
    # otherwise there is the possibility of resource leaks server side that
    # will only be cleaned up periodically.
    handle.close()

Example output:

Created Project:117 in group private-1
Created Project:118 in group read-only-1
Found 2 Project(s)
Querying data across users in the same group

Through the use of an omero.sys.ParametersI filter, restricting a query to a given user is possible. For the purposes of these examples, we will assume that both user-2 and user-3 have a single project each in the read-only-1 group.

```python
from omero.model import *
from omero.rtypes import *
from omero.sys import ParametersI

# Session that has already been created for user-2
session = client.getSession()

# Retrieve the services we are going to use
admin_service = session.getAdminService()
query_service = session.getQueryService()

# Groups we are going to query
read_only_group = admin_service.lookupGroup('read-only-1')

# Users we are going to query
user_2 = admin_service.lookupExperimenter('user-2')
user_3 = admin_service.lookupExperimenter('user-3')

# Print the members of 'read-only-1'
print 'Members of "read-only-1" (experimenter_id, username): %r' %
    [(v.id.val, v.omeName.val) for v in read_only_group.linkedExperimenterList()]

# Query for all projects
ctx = {'omero.group': str(read_only_group.id.val)}
projects = query_service.findAllByQuery('select p from Project as p', None, ctx)
print 'All projects in "read-only-1" (project_id, owner_id): %r' %
    [(v.id.val, v.details.owner.id.val) for v in projects]

# Query for projects owned by 'user-2'
ctx = {'omero.group': str(read_only_group.id.val)}
params = ParametersI()
params.addId(user_2.id.val)
projects = query_service.findAllByQuery('select p from Project as p' +
    ' where p.details.owner.id = :id', params, ctx)
print 'Projects owned by "user-2" in "read-only-1" (project_id, owner_id): %r' %
    [(v.id.val, v.details.owner.id.val) for v in projects]

# Query for projects owned by 'user-3'
ctx = {'omero.group': str(read_only_group.id.val)}
params.addId(user_3.id.val)
projects = query_service.findAllByQuery('select p from Project as p' +
    ' where p.details.owner.id = :id', params, ctx)
print 'Projects owned by "user-3" in "read-only-1" (project_id, owner_id): %r' %
    [(v.id.val, v.details.owner.id.val) for v in projects]
```

Example output:

Members of "read-only-1" (experimenter_id, username): [(10L, 'user-2'), (9L, 'user-3')]
All projects in "read-only-1" (project_id, owner_id): [(4L, 10L), (7L, 9L)]
Projects owned by "user-2" in "read-only-1" (project_id, owner_id): [(4L, 10L)]
Projects owned by "user-3" in "read-only-1" (project_id, owner_id): [(7L, 9L)]
Utilizing the Permissions object

Every object that is retrieved from the server via the query service, regardless of the context used, has a fully functional `omo.model.PermissionsI` object. This object contains various methods to allow the caller to interrogate the operations that are possible by the current user on the object:

- `canAnnotate()`
- `canDelete()`
- `canEdit()`
- `canLink()`

19.6.3 Troubleshooting permissions issues

Data disappears after a change of the primary group of a user

As outlined above, changes were made so that by default queries do not span multiple groups and the ‘Primary or Default Group’ is essentially a deprecated concept. If you have multiple groups and you are attempting to make queries by switching the ‘Active Group’ via the `setSecurityContext()` method of an active session (`omero.cmd.SessionPrx`), those queries will be scoped only to that group. If you want your queries to act more like they did in 4.1.x, setting `omero.group=-1` will achieve that.

However, the reasons we made these changes have more to them than just API usage and the OMERO client history of only showing the data from one group at a time. Changing the ‘Active Group’ is both expensive because of the atomicity requirements the server enforces and can create dangerous concurrency situations. This is further complicated by the addition of the change group and delete background processes since 4.1.x. Manipulating a session’s ‘Primary or Default Group’ during these tasks can have drastic effects. Changing the ‘Active Group’ is forbidden if there are any stateful services (`omero.api.RenderingPrx` for example) currently open.

In short, in OMERO 5.2.6 you absolutely should not be switching the ‘Primary or Default Group’ of the user, or the ‘Active Group’ of a session, as a means to achieve cross group querying.

Listing other users’ data in read-only groups

In order to list other users’ data associated with read-only groups of which you are a member, you can also use the context object and set the `omero.group` to -1. In addition, you can add a filter to the query to only select the other users’ data. You can do this either by using the `omero.sys.ParametersI` object’s `exp()` method when using the `IContainer` service, or by an explicit query when using `IQuery` service.

Is the default group the primary group when not specifying the context?

The value of the `groupId` property of the `omero.sys.EventContext` is the “Active Group” for the created session. It can be modified as described above with the restrictions outlined. Unless the session has been created by means other than `createSession()` on an `omero.client` object, this will be the user’s “Primary or Default Group.” A user’s ‘Primary or Default Group’ is the first group in the collection that describes the relation `Experimenter <--- ExperimenterGroup`. It can be set by the `setDefaultGroup()` method on the `IAdmin` service.

What about when importing data without specifying the context object?

Exactly as outlined above. Import does nothing different or special. If you want the operating context of an import to be different from the default you must specify it as such.

---

63 http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/model/Permissions.html#canAnnotate
64 http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/model/Permissions.html#canDelete
65 http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/model/Permissions.html#canEdit
66 http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/model/Permissions.html#canLink
Specifying the group context as -1 when deleting data

There is no need to do this. Complete graphs cannot span multiple groups and queries are only (unless otherwise filtered) restricted at the group level and not at the level of the user. Furthermore, the delete service always internally performs all its queries in the `omero.group=-1` context unless another more explicit one is specified.
Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

20.1 OMERO.server overview

20.1.1 OMERO sequence narrative

Trying to understand all of what goes on with the server can be a bit complicated. This short narrative tries to touch on the most critical aspects.

• A request reaches the server over one of the two remoting protocols: RMI or ICE. First, the Principal\(^1\) is examined for a valid session which was created via ISession.createSession(String username, String password)\(^2\).

• These values are checked against the experimenter, experimentergroup and password tables. A valid login consists of a user name which is to be found in the omename column of experimenter. This row from experimenter must also be contained in the “user” experimenter group which is done via the mapping table groupexperimentermap (see this SQL template\(^3\) for how root and the initial groups are setup).

• If the server is configured for LDAP Authentication, an Experimenter may be created when ISessions attempts to check the password via IAdmin.checkPassword().

• If authentication occurs, the request is passed to an EJB\(^4\) interceptor which checks whether or not the authenticated user is authorized for that service method. Methods are labelled either @RolesAllowed("user"), @RolesAllowed("system"), or @PermitAll. All users are a member of “user”, but only administrators will be able to run “system” methods.

• If authorization occurs, the request finally reaches a container-managed stateful or stateless service. The service will prepare the OMERO runtime for the particular user – checking method parameters, creating a new event, initializing the security system, etc. – and pass execution onto the method implementation. This is done using references acquired (or injected) from the Spring application context.

• The actual service implementation (from ome.logic\(^5\) or ome.services\(^6\)) will be either read-only (IQuery\(^7\)-based) or a read-write (IUpdate\(^8\)-based).

• In the case of a read-only action, the implementation asks the database layer for the needed object graph, transforms them where necessary, and returns the values to the remoting subsystem. On the client-side, the returned graph can be mapped to an internal model via the (OMERO Model Mapping|model wrapper)).

• In the case of a read-write action, the change to the database is first passed to a validation layer for extensive checking. Then the graph is passed to the database layer which prepares the SQL, including an audit trail of the changes made to the database.

\(^1\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/system/Principal.java
\(^2\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/ISession.java
\(^3\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/dsl/resources/ome/dsl/psql-footer.vm
\(^4\)http://www.oracle.com/technetwork/java/javasee/jjb/index.html
\(^5\)https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/server/src/ome/logic
\(^6\)https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/server/src/ome/services
\(^7\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/IQuery.java
\(^8\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/IUpdate.java
• After execution, the OMERO runtime is reset, the method call is logged, and either the successful results are returned or an exception is thrown.

20.1.2 Technologies

It is fairly easy to work with the server without understanding all of its layers. The API is clearly outlined in the ome.api package and the client proxies work almost as if the calls were being made from within the same virtual machine. The only current caveat is that objects returned between two different calls will not be referentially (i.e. \( \text{obj1} == \text{obj2} \)) equivalent. We are working on removing this restriction.

To understand the full technology stack, however, there are several concepts which are of importance:

• A layered architecture ensures that components only “talk to” the minimum necessary number of other components. This reduces the complexity of the entire system. Ensuring a loose-coupling of various components is facilitated by dependency injection. Dependency injection is the process of allowing a managing component to place a needed resource in a component’s hand. Code for lookup or creation of resources, in turn, is unneeded, and explicit implementation details do not need to be hard-coded.

• Object-relational mapping (ORM) is the process of mapping relational tables to object-oriented classes. Currently OMERO uses Hibernate\(^9\) to provide this functionality. ORM allows the developer to work in a known environment, here the type-safe world of Java, rather than writing difficult to debug sql.

• Aspect-oriented programming, a somewhat new and misunderstood technology, is perhaps the last technology which should be mentioned. Various pieces of code (“aspects”) are inserted at various moments (“joinpoints”) of execution. Collecting logic into aspects, whether logging, transactions, security etc., also reduces the overall complexity of the code.

20.1.3 Server design

The server logic resides in the components/server\(^10\) component.

Topics

• Exception handling
• OME-Remote Objects
• Server security and firewalls

See also:

OMERO.grid

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

20.2 Extending OMERO.server

Overview

Despite all the effort put into building OMERO, it will never satisfy the requirements of every group. Where we have seen it useful to do so, we have created extension points which can be used by third-party developers to extend, improve, and adapt OMERO. We outline most of these options below as well as some of their trade-offs. We are also always interested to hear other possible extension points. Please contact the ome-devel mailing list\(^*\) with any such suggestions.

\(^9\)http://www.hibernate.org
\(^10\)https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/server

\(^*\)http://lists.openmicroscopy.org.uk/mailman/listinfo/ome-devel/
20.2.1 Existing extension points

To get a feeling for what type of extension points are available, you might want to take a look at the following pages. Many of them will point you back to this page for packaging and deploying your new code.

- **File parsers** - write Java file parsers to further extend search
- **LoginAttemptListener** - write a Java handler for failed login attempts
- **Command Line Interface as an OMERO development tool** - write drop in Python extensions for the command-line
- **Introduction to OMERO.scripts** - write python scripts to process data server-side
- **LDAP plugin design** - write a Java authentication plugin
- **Password Provider** - write a Java password backend
- **Search bridges** - write Java Lucene parsers to extend search
20.2. Extending OMERO.server

- **OMERO.mail** - server email sender (added in OMERO 5.1, no developer documentation as yet)
- **omero.policy.bean** - policy configuration point e.g. for setting download restriction policy on users (added in OMERO 5.1, no developer documentation as yet)
20.2.2 Extending the Model

The OME Data Model and its OMERO representation, the OME-Remote Objects, intentionally draw lines between what metadata can be supported and what cannot. Though we are always examining new fields for inclusion, it is not possible to represent everyone’s model within OME.

Structured annotations

The primary extension point for including external data are the Structured annotations (SAs). SAs are designed as email-like attachments which can be associated with various core metadata types. In general, they should link to information outside of the OME model, i.e. information which OMERO clients and servers do not understand. URLs can point to external data sources, or XML in a non-OME namespace can be attached.

The primary drawbacks are that the attachments are opaque and cannot be used in a fine-grain manner.

Code generation

Since it is prohibitive to model full objects with the SAs, one alternative is to add types directly to the generated code. By adding a file named *-ome.xml to components/model/resources/mappings\(^{11}\) and running a full-build, it is possible to have new objects generated in all OMERO.blitz languages. Supported fields include:

- boolean
- string
- long
- double
- timestamp
- links to any other ome.model.* object, including enumerations

For example:

```
<types>
  <!-- "named" and "described" are short-cuts to generate the fields "name" and "description" -->
  <type id="ome.model.myextensions.Example" named="true" described="true">
    <required name="valueA" type="boolean"/> <!-- This is NONNULL -->
    <optional name="valueB" type="long"/>  <!-- This is nullable -->
    <onemany name="images" type="ome.model.core.Image"/> <!-- A set of images -->
  </type>
</types>
```

Collections of primitive values like `<onemany name="values" type="long"/>` are not supported. Please see the existing mapping files for more examples of what can be done.

The primary drawback of code-generating your own types is isolation and maintenance. Firstly, your installation becomes isolated from the rest of the OME ecosystem. New types are not understood by other servers and clients, and cannot be exported or shared. Secondly, you will need to maintain your own server and client builds of the system, since the provided binary builds would not have your new types.

Measurement results

For storing large quantities of only partially structured data, such as tabular/CSV data with no pre-defined columns, neither the SAs nor the code-generation extensions are ideal. SAs cannot easily be aggregated, and code-generation would generate too many types. This is particularly clear in the storage and management of HCS analysis results.

To solve this problem, we provide the OMERO.tables API for storing tabular data indexed via Roi, Well, or Image id.

\(^{11}\)https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/model/resources/mappings
20.2.3 Services

Traditionally, services were added via Java interfaces in the components/common/src/ome/api\(^{12}\) package. The creation of such “core” services is described under *How To create a service*. However, with the introduction of *OMERO.blitz*, it is also possible to write blitz-only services which are defined by a slice definition under components/blitz/resources/omero\(^{13}\).

A core service is required when server internal code should also make use of the interface. Since this is very rarely the case for third-party developers wanting to extend OMERO, only the creation of blitz services will be discussed here.

**Add a slice definition**

The easiest possible service definition in slice is:

```plaintext
module example {
    interface NewService {
        void doSomething();
    };
};
```

This should be added to any existing or a new *.ice file under the blitz/resources/omero directory. After the next ant build, stubs will be created for all the *OMERO.blitz* languages, i.e. *OMERO Java language bindings*, *OMERO Python language bindings*, and *OMERO C++ language bindings*.

**Note:** Once you have gotten your code working, it is most reusable if you can put it all in a single directory under tools/. These components also have their resources/*.ice files turned into code, and they can produce their own artifacts which you can distribute without modifying the main code base.

**Warning: exceptions**

You will need to think carefully about what exceptions to handle. Ice (especially *OMERO C++ language bindings*) does not handle exceptions well that are not strictly defined. In general, if you would like to add your own exception type, feel free to do so, but either 1) subclass `omero::ServerError` or 2) add to the appropriate throws clauses. And regardless, if you are accessing any internal OMERO API, add `omero::ServerError` to your throws clause.

See *Exception handling* for more information.

**Java implementation using _Disp**

To implement your service, create a class subclassing “example._NewServiceDisp” class which was code-generated. In this example, the class would be named “NewServiceI” by convention. If this service needs to make use of any of the internal API, it should do so via dependency injection. For example, to use IQuery add either:

```java
void setLocalQuery(LocalQuery query) {
    this.query = query;
}
```

or

```java
NewServiceI(LocalQuery query) {
    this.query = query;
}
```

The next step “Java Configuration” will take care of how those objects get injected.

---

\(^{12}\)https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/common/src/ome/api

\(^{13}\)https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/blitz/resources/omero
Java implementation using _Tie

Rather than subclassing the _Disp object, it is also possible to implement the _Tie interface for your new service. This allows wrapping and testing your implementation more easily at the cost of a little indirection. You can see how such an object is configured in blitz-servantDefinitions14.

Java configuration

Configuration in the Java servers takes place via Spring15. One or more files matching a pattern like ome/services/blitz-*.xml should be added to your application.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">
<beans>
  <bean class="NewServiceI">
    <description>
      This is a simple bean definition in Spring. The description is not necessary.
    </description>
    <constructor-arg ref="internal-ome.api.IQuery"/>
  </bean>
</beans>
```

The three patterns which are available are:

- ome/services/blitz-*.xml - highest-level objects which have access to all the other defined objects.
- ome/services/services-*.xml - internal server objects which do not have access to blitz-*.xml objects.
- ome/services/db-*.xml - base connection and security objects. These will be included in background java process like the index and pixeldata handlers.

**Note:** Password Provider and similar should be included at this level.

See components/blitz/resources/ome/services16 and components/server/resources/ome/services17 for all the available objects.

Java deployment

Finally, these resources should all be added to OMERO_DIST/lib/server/extensions.jar:

- the code generated classes
- your NewServiceI.class file and any related classes
- your ome/service/blitz-*.xml file (or other XML)

Non-service beans

In addition to writing your own services, the instructions above can be used to package any Spring-bean into the OMERO server. For example:

```java
//
// MyLoginAttemptListener.java
//
import ome.services.messages>LoginAttemptMessage;
```

---

14https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/resources/ome/services/blitz-servantDefinitions.xml#L36
15http://spring.io
16https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/blitz/resources/ome/services
17https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/server/resources/ome/services
import org.springframework.context.ApplicationListener;

/**
 * Trivial listener for login attempts.
 */
public class MyLoginAttemptListener implements ApplicationListener<LoginAttemptMessage> {
    public void onApplicationEvent(LoginAttemptMessage lam) {
        if (lam.success != null && !lam.success) {
            // Do something
        }
    }
}

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">
<beans>
    <bean class="myLoginAttemptListener" class="MyLoginAttemptListener">
        <description>
            This listener will be added to the Spring runtime and listen for all LoginAttemptMessages.
        </description>
    </bean>
</beans>

Putting MyLoginAttemptListener.class and ome/services/blitz-myLoginListener.xml into lib/server/extensions.jar is enough to activate your code:

~$ ls -l
MyLoginListener.class
MyLoginListener.java
lib
...
~$ jar cvf lib/server/extensions.jar MyLoginListener.class ome/services/blitz-myLoginListener.xml
added manifest
adding: MyLoginListener.class {in=0} {out=0} {stored 0%}
adding: ome/services/blitz-myLoginListener.xml {in=0} {out=0} {stored 0%}

20.2.4 Servers

With the OMERO.grid infrastructure, it is possible to have your own processes managed by the OMERO infrastructure. For example, at some sites, Nginx\(^{18}\) is started to host OMERO.web framework. Better integration is possible however, if your server also uses the Ice\(^{19}\) remoting framework.

One way or the other, to have your server started, monitored, and eventually shutdown by OMERO.grid, you will need to add it to the “application descriptor” for your site. When using:

\(^{18}\)http://wiki.nginx.org/Main
\(^{19}\)https://zeroc.com
the application descriptor used is etc/grid/default.xml. The <application> element contains various <node>s. Each node is a single daemon process that can start and stop other processes. Inside the nodes, you can either directly add a <server> element, or in order to reuse your description, you can use a <server-instance> which must refer to a <server-template>.

To clarify with an example, if you have a simple application which should watch for newly created Images and send you an email: mail_on_import.py, you could add this in either of the following ways:

**Server element**

```
<node name="my-emailer-node"> <!-- this could also be an existing node, but it must be unique -->
  <server id="my-emailer-server" exe="/home/josh/mail_on_import.py" activation="always">
    <env>${PYTHONPATH}</env>
    <!-- The adapter name must also be unique -->
    <adapter name="MyAdapter" register-process="true" endpoints="tcp" />
  </server>
</node>
```

**Server-template and server-instance elements**

```
<server-template id="emailer-template"> <!-- must also be unique -->
  <property name="user" />
  <server id="emailer-server-${user}" exe="/home/${user}/mail_on_import.py" activation="always">
    <env>${PYTHONPATH}</env>
    <!-- The adapter name must also be unique -->
    <adapter name="MyAdapter" register-process="true" endpoints="tcp" />
  </server>
</server-template>

<node name="our-emailer-node">
  <server-instance id="emailer-template" user="ann">
  </server-instance>
</node>
```

See also:

[ome-devel] model description driven code generation

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

## 20.3 OMERO.blitz

The OMERO.blitz server is responsible for providing secure access to data and metadata via user sessions (OMERO sessions), and cleaning up all resources when they are no longer being used. Various server capabilities are accessed via a multitude of services collectively known as the OMERO Application Programming Interface.

### 20.3.1 Metadata

Metadata stored in an object-relational database is mapped into the OMERO OME-Remote Objects via Hibernate. Hibernate Query Language (HQL) calls can be made against the server and have all ownership information automatically taken into account.

---

20.3.2 Image data

The binary image data can either be accessed in its raw form via the RawPixelsStore service, or can be rendered by the OMERO.server image rendering service.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

20.4 OMERO.fs

OMERO.fs is a series of on-going changes designed to improve the way an OMERO.server interacts with existing directories of acquired image data. It currently consists of two components:

OMERO.dropbox is designed for watching a directory and kicking off an automatic import. The configuration of the DropBox system is covered on the OMERO.dropbox system administrator’s page.

OMERO.fs Managed Repository is designed to store original data in an unaltered form without requiring the data duplication that was carried out by a pre-5.0 import. The changes to the import system mean that an OMERO.fs server stores the original files in the ManagedRepository, preserving file names and any nested directory structure. The repository may even contain direct in-place links to original data without an file upload step. These changes improve the way OMERO deals with High Content Screening (HCS) and other complex heterogeneous data types, reducing storage requirements and using Bio-Formats to recognize Filesets (groups of files which correspond to multi-dimensional images and accompanying information) so they can be treated as single entities within the OMERO clients. OMERO.fs has some configuration properties that allow sysadmins to customize it for their site; developers should be aware of the how the new import process works and how this is affected by the configuration.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

20.5 Import under OMERO.fs

The OMERO 5 release introduces OMERO.fs, a new way of storing files in the OMERO binary repository and thus a new method of importing images to the server.

In previous versions of OMERO the import process was very much client-centered. When importing an image the client pushed pixel data, metadata and, optionally, original image files to the OMERO server. With the advent of OMERO 5, OMERO.fs allows the pixels to be accessed directly from the original image files. This means that much of the import process can now take place on the server once the original image files have been uploaded.

This page looks at the implications for the developer writing import clients. A broad description of the import workflow is followed by some of the model changes needed to facilitate this workflow. The current API sequence is then introduced before looking at server-side classes and sequence. Finally, the configuration required for OMERO.fs is specified.

20.5.1 Import overview

The broad import workflow comprises selecting a file or set of files to be imported client-side. Using Bio-Formats on the client this selection is resolved into a number of import candidates. Here an import candidate is a file or set of files that represents a single image, a multi-image set or a plate. Each import candidate, which may be one file or several files, is then treated as Fileset for import. The import of each Fileset is then undertaken by the client in two stages: upload and server-side import.

A Fileset is uploaded to the server into a location determined by the server, multiple Filesets may be uploaded in parallel by a client. A checksum is calculated before upload by the client and after upload by the server. If these checksums match then the client triggers a server-side import. The client can then move on to doing other work and leave the import to complete on the server.

Once the Fileset is on the server and an import has been initiated by the client all processing then takes place on the server. The server then uses Bio-Formats to extract and store the metadata, calculate the minimum and maximum pixel values and do other import processing.
20.5.2 Filesets

A fileset is a concept new to OMERO.fs which captures how Bio-Formats relates files to the images that they encode. If importing a single file leads to a single corresponding image being viewable then a one-to-one mapping exists from file to image. However, an image's information may be split among multiple files, or a single file may encode multiple images. In other cases, especially in high-content screening, many images of wells may be encoded in a complex manner by many files. In all these cases, a fileset is used to hold the set of files and the set of images to which those files correspond.

http://www.openmicroscopy.org/site/support/bio-formats/
Filesets are essentially indivisible: the files or images that a fileset represents are deleted, or moved between groups, only as one unit together, and on the server each fileset has a directory in which only its files are stored. Each fileset firmly binds sets of files and images because the dependencies among them mean that splitting components away leaves a partial fileset making the remaining data unreadable. Similarly, you should not rename any components of a fileset as this will also break the dependencies holding them together.

### 20.5.3 Model description

See acquisition.ome.xml[23].

- **ome.model.fs.Fileset**
  - Represents a group of files which are considered to belong together.
  - In the future the client may upload a single “import set” that is subsequently resolved by Bio-Formats into multiple filesets on the server.
  - Also links to multiple ome.model.jobs.Job instances.
  - Links to the image objects that are created during import.
- **ome.model.jobs.Job subclasses**
  - Each one represents some action which takes place server-side on the Fileset.
  - For the standard sequence described above, the first will always be an UploadJob which contains version info from the client. If the files were not uploaded however, this may not be the case. Then a MetadataImportJob follows, which is the basic import.
  - Other jobs may be necessary for regular usage (PixelDataJob, IndexingJob, etc.). *Later jobs may also be added like a re-parse job, a re-check of the hashes to detect corruption, or an archive job.*
  - For job definitions, see jobs.ome.xml[24].
  - Some subclasses have a versionInfo property for storing a snapshot of process information along with software versions. Most important for knowing how files were parsed, therefore when using importPaths, a “synthetic” version info might be created to say that these were just uploaded blindly.
- **ome.model.fs.FilesetEntry**
  - Link from a Fileset to exactly one ome.model.core.OriginalFile
  - Critically, it also contains the original absolute client path of that file.

### 20.5.4 API sequence

- Choose which files to import by either:
  - **ImportLibrary** and friends (Java only)
  - listing all files (not directories) manually.
- Choose a **ManagedRepositoryPrx** from SharedResourcesPrx.repositories().
- Call either:
  - **ImportLibrary.importImage()** which calls ManagedRepositoryPrx.importFilesets(Fileset, ImportSettings), or
  - directly use ManagedRepositoryPrx.importPaths(StringSet).
- Receive a **ImportProcessPrx**.
- For each **FileEntry** in the FileSet or each path in the StringSet (in order), call **ImportProcessPrx.getUploader()** and receive a **RawFileStorePrx**.
- Upload the file via **RawFileStorePrx.write()** while reading the files locally to write, be sure to calculate the checksum.

[23]https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/model/resources/mappings/acquisition.ome.xml
• Pass a list of checksums (in order) to ImportProcessPrx.verifyUpload(StringSet). If the hashes match, receive a HandlePrx. Otherwise an exception is thrown.

At this point, the client should be able to disconnect and the rest of the import should happen independently.

• Create a CmdCallbackI that wraps the HandlePrx and wait for successful completion.

At this point, the main metadata import is finished, but background processing may still be occurring. Handles for the background processing will also be returned.

20.5.5 Server-side classes/concepts

AbstractRepositoryI and all of its subclasses are implementations of the InternalRepository API. These objects are for internal use only and should never be accessible by the clients. Each instance is initialized with a directory which the servant attempts to “acquire” (i.e. grab a lock file). Once it does so, it is the serving repository.

Each internal repository provides a public view which in turn provides the Repository API. All method calls assume Unix-style strings, which are guaranteed by CheckedPath, a loose wrapper around java.io.File. CheckedPath objects along with the active Ice.Current instance are passed to the RepositoryDao interface, which provides database access for all repositories. Access to raw IO is provided by the RepoRawFileStoreI servant, which wraps a RawFileBean.

The ManagedRepository implementation is responsible for import and enforces further constraints (beyond those of CheckedPath) on where and what files are created. Most importantly, the omero.fs.repo.path template value is expanded and prepended to all uploads. A further responsibility of the ManagedRepository is to maintain a list of all currently running ManagedImportProcessI, each of which is held in the ProcessContainer. These ManagedImportProcessI instances further wrap RepoRawFileStoreI instances with a subclass, ManagedRawFileStoreI. For file import through ManagedRepository.importFileset, although hasher is nullable ordinarily, it will be set through the mandatory ImportSettings.checksumAlgorithm property. ManagedRepository.listChecksumAlgorithm lists the hashers supported by the server. ManagedRepository.suggestChecksumAlgorithm helps the client and server to negotiate a mutually acceptable algorithm, as in ImportLibrary.createImport; the result is affected by the server’s configuration setting for omero.checksum.supported. ImportLibrary calculates each file’s hash using hashers obtained through ChecksumProviderFactory.getProvider. In fetching OriginalFile objects by HQL through the Query Service one needs JOIN FETCH on the hasher property to read the hasher’s name.

20.5.6 Server-side sequence

NB: Server-side ImportLibrary is no longer being used. That logic is currently moved to ManagedImportRequestI. This may not be the best location. Further, several other layers might also be collapsible, like OMEROMetadataStore which is currently accessible as a “hidden” service MetadataStorePrx. Here, “hidden” means that it is not directly retrievable from ServiceFactoryPrx.

• ManagedRepositoryI.importPaths()
  – reuses ImportContainer.fillData() to create an ImportSettings and a Fileset and then calls importFileset(Fileset, ImportSettings)

• ManagedRepositoryI.importFileset()
  – determines an ImportLocation calling PublicRepositoryI.makeDir() where necessary.
  – createImportProcess creates a ManagedImportProcessI, registers it, and returns it.
  – After this, the repository is only responsible for periodically having the ping and eventually the shutdown method called, via ProcessContainer.

• ManagedImportProcessI.getUploader()
  – creates a new RepositoryRawFileStoreI for each file in the paths/fileset.
  – Once close() is called on this instance, closeCalled(int i) will be called on the import process and the instance will be removed.
  – If getUploader() is called again, then a new file store will be created.

• ManagedImportProcessI.verifyUpload()
If all hashes match, then a ManagedImportRequestI instance is created and submitted to omero.cmd.SessionI.submit_async() for background processing. The client can wait for the returned omero.cmd.HandlePrx to finish by using a CmdCallbackI.

At this point, the ImportProcessPrx can be closed as well as the entire client and the import would still continue. Only if HandlePrx.cancel() is called, will the import be aborted.

QUESTION: How to handle rollback at this point?

* ManagedImportRequestI.init (within transaction)
  * Registry.getInternalServiceFactory() grabs a ServiceFactoryPrx without the need for an omero.client instance.
  * OMEROWrapper and a OMEROMetadataStoreClient are created with this connection.
  * Some other basic configuration takes place.

* ManagedImportRequest.step() (N times, each within same transaction as init())
  * NB: it may later make more sense for this bit to happen in a separate process.
  * At the moment, 5 steps are hard-coded. Each performing roughly the same amount of work. Some of these may later be done in the background.
    * importMetadata() calls store.saveToDB(), which calls MetadataStorePrx.saveToDb(), a remote call. This could possibly be inlined.
    * generateThumbnails() calls store.resetDefaultsAndGenerateThumbnails(), another remote call, which could also be inlined.
    * pixelData() calls store.setPixelsParams(), updatePixels(), and populateMinMax(). Min/Max especially should be backgrounded.
    * Finally, store.launchProcessing() is called, which should remain, but could also be inlined. The returned script processes could be returned in the ImportResponse.
  * Return appropriate values.

* notifyObservers() currently does nothing, since this was client-side functionality in ImportLibrary. This needs to be replaced!

* ManagedImportRequest.buildResponse() (N times, outside the transaction)
  * Only step 4 does anything, storing the pixels in a ImportResponse

* ManagedImportRequest.getResponse() (1 time, regardless of exception or not)
  * Performs cleanup, then returns the ImportResponse assuming that no call to helper.cancel() has been made. At this point, ImportLibrary.importImage() returns successfully.

See also:

FS configuration options

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

## 20.6 OMERO.processor

The Processor is a python process-launcher which can be run on any Unix system to execute scripts for a user. This makes use of the scripting service functionality. As many processor nodes can be started as physical computers are available.

* Source code: components/tools/OmeroPy/src/omero/processor.py

* Documentation: OMERO.grid

25https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/tools/OmeroPy/src/omero/processor.py
20.7 OMERO.server image rendering

A major requirement for any image data application is the ability to display images. In most applications, this is achieved by reading pixel data from a filesystem and then mapping the pixel data to the 256 grey level available on most computer display monitors. It is common in some experiments to record and display multiple channels at once. Typically three, four, or even five separate images must be mapped, and then presented as a color image for painting on a monitor. Because these operations can require many thousands of operations and must be displayed rapidly to support the display of time-lapse movies, most image display software applications use a high-speed graphics CPU and dedicated hardware for image rendering and display. This requirement limits the deployment of these applications to high-powered workstations.

OMERO.server includes an image server, a software application that delivers rendered images to a client. This ensures that client applications can display image data. The OMERO Rendering Engine (OMERO-RE) has been designed to minimize the amount of data transferred to the client and thus removes the requirement for a specific graphics CPU, allowing high-performance image viewing on standard laptop computers. The OMERO-RE achieves this by limiting data transfer times by being close to the data, using highly efficient network transfer protocols, utilizing modern multi-processor and multi-core machines to provide the data to clients in a format that is as efficient to display as possible. OMERO-RE is multi-threaded and can use multi-core servers to simultaneously render individual channels before assembly into a final color image ready for transfer to the client. The use of the RE is not mandatory. If a client needs to have the full pixel data, it can. This OriginalPixels facility is used for client-side analysis, like that performed in the OMERO.insight measurement tool.

Transfer of image data even after rendering can limit performance, especially when accessing data remotely on connections with limited bandwidth (e.g. domestic ADSL). Therefore the OMERO-RE contains a compression service with an API that allows a client adjustable compression providing minimal image artefacts and a 20-fold range of data size to the client.

The OMERO Rendering Engine is accessed by OMERO client applications written in Java, C++, or Python via a binary protocol (ICE) provided by ZeroC\textsuperscript{26}.

20.8 Clustering

Clustering an OMERO instance consists of starting multiple OMERO.blitz servers with each allocating user sessions based on some criteria. There are at least two reasons you may want to cluster the OMERO server: availability and throughput.

20.8.1 Availability

Having the ability to have two servers up at the same time implies that even if you have to restart one of the servers, there should be no down-time. Currently, OMERO sessions are sticky to a cluster node and so it is not possible shutdown a node at any time. All new sessions can be redirected to the server which is to be left turned on however, then when all active sessions have completed, the chosen server can be shutdown.

20.8.2 Throughput

The other main reason to have other servers running is to service more user sessions simultaneously. Out of the box, each OMERO.blitz process is configured for 400MB of memory. When dealing with memory intensive operations like rendering, each added server can make a positive difference. This is only a part of the story, since much of the bottleneck is not the server itself but other shared resources, like the database or the filesystem, and so to further extend throughput, you will need to parallelize these.

\textsuperscript{26}https://zeroc.com
20.8.3 Installation

If you are using the default OMERO.grid application descriptor\(^{27}\) quickly enabling clustering is as simple as executing:

```
bin/omero config set omero.cluster.redirector configRedirector
bin/omero node backup start
```

This starts a second node, named “backup”, which contains a second OMERO.blitz server, “Blitz-1”. By default, this newly created server will not be used until sessions are manually redirected to it.

**See also:**

*Scaling Omero*

---

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

20.9 Collection counts

The IContainer\(^{28}\) interface has always provided a method for returning the count of some collection types via `getDetails().getCounts()` . The server has database views for all link collections. These are accessed through HQL directly, such as:

```java
Long self = iAdmin.getEventContext().getCurrentUserId();
Image i = iQuery.findByQuery("select i from Image i left outer join fetch i.annotationLinksCountPerOwner", null);
Map<Long, Long> countsPerOwner = i.getAnnotationLinksCountPerOwner();

// Map may be null if not fetched.
if (countsPerOwner != null) {
    // countOfAnnotationsForImageByUser
    Long count = countsPerOwner.get(self);
    if (count != null) {
        // do something
    }
}
```

Values written to the map will not be persisted to the database, since they are continually re-generated.

20.9.1 Pojo options

The PojoOptions configuration of what elements are counted has been removed from the API. Instead, the returned map contains all values for all users, and can be summed to acquire the total count.

20.9.2 Restrictions

Currently a Hibernate bug (waiting to be filed) prevents retrieving the counts on any other than the top-level object (“select this”).

20.9.3 Instructions

The views.sql script is automatically executed when initializing your database.

---

\(^{27}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/etc/templates/grid/default.xml

\(^{28}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/IContainer.java

20.9. Collection counts
of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

## 20.10 How To create a service

### Overview

These instructions are for core developers only and may be slightly out of date. They will eventually be revised, but if you are looking for general instructions on extending OMERO with a service, see *Extending OMERO.server*. If you would indeed like to create a core service, please contact the [ome-devel mailing list](http://www.openmicroscopy.org/site/community/mailing-lists)

To fulfill #306, r905 provides all the classes and modifications needed to create a new stateless service (where this varies from stateful services is also detailed). In brief, a service provider must create an interface, an implementation of that interface, a Spring configuration file, as well as modify the server configuration and the central service factory. (These last two points stand to change with #314).

**Note:** With the creation of *OMERO.blitz*, there are several other locations which need to be modified. These are also listed below.

### 20.10.1 Files to create

- **components/common/src/ome/api/IConfig.java**: the interface which will be made available to client and server alike (which is why all interfaces must be located in the /common component). Only serializable and client-available types should enter or exit the API. Must subclass `ome.api.ServiceInterface`

- **components/server/src/ome/logic/ConfigImpl.java**: the implementation which will usually subclass `AbstractLevel1,2Service` or `AbstractBean` (See more below on super-classes) This is class obviously requires the most work, both to fulfill the interface’s contract and to provide all the metadata (annotations) necessary to properly deploy the service.

- **components/server/resources/ome/services/service-ome.api.IConfig.xml**: a Spring configuration file, which can “inject” any value available in the server (Omero)context into the implementation. Two short definitions are the minimum. (Currently not definable with annotations.) As explained in the file, the name of the file is not required and in fact the two definitions can be added to any of the files which fall within the lookup definition in the server’s `beanRefContext.xml` file (see below).

- **components/blitz/src/ome/services/blitz/impl/ConfigI.java**: an Ice “servant” implementation which can use on of several methods for delegating to the `ome.api.IConfig` interface, but all of which support *throttling*.

### 20.10.2 Files to edit (not strictly necessary, see #314)

- **components/common/src/ome/system/ServiceFactory.java**: our central API factory, needs an additional method for looking up the new interface (`getInterface name>Service()`)

- **components/server/resources/ome/services/server Spring configurations**: which makes the use of JNDI and JAAS significantly simpler.

### References

- [https://trac.openmicroscopy.org/ome/ticket/306](https://trac.openmicroscopy.org/ome/ticket/306)
- [https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/IConfig.java](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/src/ome/api/IConfig.java)
- [https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/logic/ConfigImpl.java](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/logic/ConfigImpl.java)
- [https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/resources/ome/services/service-ome.api.IConfig.xml](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/resources/ome/services/service-ome.api.IConfig.xml)
- [https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/resources/beanRefContext.xml](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/resources/beanRefContext.xml)
- [https://trac.openmicroscopy.org/ome/ticket/314](https://trac.openmicroscopy.org/ome/ticket/314)
- [http://spring.io](http://spring.io)
- [https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/resources/beanRefContext.xml](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/resources/beanRefContext.xml)
- [https://zeroc.com](https://zeroc.com)
components/blitz/resources/omero/API.ice (blitz) a ZeroC slice definition file, which provides cross-language mappings. Add the same service method to ServiceFactoryI as to ServiceFactory.java.

components/blitz/resources/ome/services/blitz-servantDefinitions.xml (blitz) a Spring configuration, which defines a mapping between Ice servants and Java services.

components/blitz/resources/omero/Constants.ice (blitz) a ZeroC slice definition file, which provides constants needed for looking up services, etc.

components/blitz/src/ome/services/blitz/impl/ServiceFactoryI.java (blitz) the central session in a blitz. Should always be edited parallel to ServiceFactory.java. Also optional in that MyServicePrxHelper.uncheckedCast( serviceFactoryI.getByName(String) ) can be used instead.

20.10.3 Files involved

components/server/resources/beanRefContext.xml

components/blitz/resources/beanRefContext.xml Singleton definitions which allow for the static location of the active context. These do not need to be edited, but in the case of the server beanRefContext.xml, it does define which files will be used to create the new context (of importance is the line classpath*:ome/services/service-*.xml). blitz’s beanRefContext.xml defines the pattern classpath*:ome/services/blitz-*.xml to allow for blitz-specific configuration.

20.10.4 And do not forget the tests

components/server/test/ome/server/itests/ConfigTest.java tests only the implementation without a container.

blitz: Currently, testing blitz is outside the scope of this document.

20.10.5 Things to be aware of

Local APIs

Several services implement a server-side subclass of the ome.api interface rather than the interface itself. These interfaces are typically in ome.api.local. Such local interfaces can provide methods that should not be made available to clients, but which are needed within the server. Though not currently used, the @Local() annotation on the implementation can list the local interface for future use. See UpdateImpl for an example.

Stateful services

Currently all stateful services are in their own component (components/rendering and components/romio) but their interface will still need to be under components/common for them to be accessible to clients. To be done.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

20.11 OMERO sessions

OMERO sessions simplifies the handling of login sessions for OMERO.blitz.

20.11. OMERO sessions
OMERO.blitz Session Creation and Destruction

OMERO Documentation, Release 5.2.6

Total of 3 ServiceFactories, for 3 clients, but single session which can only be removed if either:
1) close() is called when refCount == 0
2) session timeout
The possibly dangerous "closeOnDestroy()" (in RED) only reduces the reference count.

20.11. OMERO sessions
In short:

- Sessions are a replacement for the standard JavaEE security infrastructure.
- Sessions unify the Blitz and RMI session handling, making working with Java RMI more like Blitz (since the JavaEE interaction is essentially “conversationless”).
- Sessions provide the ability (especially in Blitz) to quit a session and rejoin it later as long as it has not timed out, possibly useful for moving from one machine to another.
- Sessions provide the ability to share the same space. Two users/clients attached to the same session would experience the same life-cycle.
- Sessions provide a scratch space to which any data can be written for and by job/script executions.
- Sessions act as a global cache (in memory or on disk) to speed up various server tasks, including login. With further extensions like http://terracotta.org/, sessions could serve as a “distributed” cache.
- Sessions prevent sending passwords in plain text or any other form. After that, all session interactions take place via a shared secret key.

20.11.1 Design

All services other than ISession, assume that a user is logging in with a username equal to session uuid. Whereas previously one logged in with:

```java
ome.system.Principal p = new ome.system.Principal("josh","user","User");
```

behind the scenes, now the “josh” value is replaced by the UUID of a ome.model.meta.Session instance. The session is acquired by a call to:

```java
ome.api.ISession.createSession(Principal principal, String credentials);
```

and carries information related to the current user’s session.

```java
Session session;
session.getUuid();  // Unique identifier; functions as a temporary password. DO NOT SHARE IT.
session.getTimeToIdle();  // Number of milliseconds which the user can idle without session timeout
session.getTimeToLive();  // Total number of milliseconds for which the session can live
session.getStarted();  // Start of session
session.getClosed();  // if != null, then session is closed
```

These properties cannot be modified.

Other properties are for use by clients:

```java
session.getMessage();  // General purpose message statement
session.getAgent();  // Can be used to specify which program the user is using
session.getDefaultEventType();  // Default event type (the third argument "User" to Principal above)
session.getDefaultPermissions();  // String representation of umask (e.g. "rw----")
```

After changing a property on the session returned by createSession() it is possible to save them to the server via:

```java
ome.api.ISession.updateSession(Session);
```

Finally, when finished, to conserve resources it is possible to destroy the session via:
20.11.2 Existing sessions

In **OMERO.blitz**, it is possible to reacquire the session if it is still active, by passing the previous session UUID as your password (User principal is ignored).

```python
client = omero.client()
servicefactory = client.createSession()
iadmin = servicefactory.getAdminService()
olduuid = iadmin.getEventContext().sessionId

// lose connection
client = omero.client()
servicefactory = client.createSession(omero.sys.Principal(), olduuid)
// now reattached
```

See also:

*Server security and firewalls*

---

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

---

20.12 Aspect-oriented programming

Aspect-oriented programming is, among other things, the attempt to define and centralize cross-cutting concerns. In other words, it is not much more than the tried-and-true principle of modularization. Having possibly unseen aspects operating on a given class however, can complicate an initial examination of the code. Therefore, it is important to be aware of what portions of the OMERO codebase are “advised” and where to find the advisors (in the case of OMERO solely interceptors).

In **Spring**[^63], advisors are declared in the bean definition files (under `components/server/resources/ome/services`, `services.xml`, `hibernate.xml`, and others.

In these configuration files, various Spring beans (shared objects) are defined with names like “proxyHandler”, “eventHandler”, “serviceHandler”, and “transactionHandler”. Each of these is a method interceptor which is passed execution before the actual logic is reached. The interceptor can inspect or replace the return value, but can also stop the method execution from ever taking place.

Unlike with **AspectJ**[^67], the AOP implementation used by OMERO only allows for the advising of interfaces. Simply creating a new service implementation via “new QueryImpl()” will not produce an advised object, which in turn will not function properly, if at all. Instead, advised objects can only be acquired from the Spring context.

By and large, only the **API service methods** are advised in OMERO.

### 20.12.1 Why?

Often, when implementing or adding code, it becomes clear just how many requirements are placed by libraries, the application server, and existing code on any new code. This can include transaction handling, session handling, security checks, object validation, logging etc. As a code-base grows, these dependencies slow development and make code unmanageable. AOP tries to reduce these dependencies by defining each of these concerns in a single place.

[^63]: http://spring.io
[^64]: https://github.com/openmicroscopy/openmicroscopy/tree/v5.2.6/components/server/resources/ome/services
[^65]: https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/resources/ome/services/services.xml
[^66]: https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/resources/ome/services/hibernate.xml
[^67]: http://eclipse.org/aspectj/
As a quick example, in OMERO transactions and exceptions are handled through method interceptors. Rather than writing:

```java
void method1()
{
    try {
        Transaction tx = new Transaction();
        tx.begin();
        // your code goes here
        tx.commit();
    } catch (TxException e) {
        tx.rollback();
    } catch (OtherException e) {
    }
}
```

you just write:

```java
void method1()
{
    // your code goes here
}
```

See also:
- [Aspect Oriented Programming](http://spring.io) Chapter of the Spring documentation
- [AOP Alliance](http://www.martinfowler.com/eaaCatalog/unitOfWork.html) Joint project defining interfaces for various AOP implementations

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

### 20.13 OmeroContext

The entire OMERO application (on a single JVM) resides in a single `ome.system.OmeroContext`. Each call belongs additionally to a single `org.hibernate.Session` (which can span over multiple calls) and to a single `ome.model.meta.Event` (which is restricted to a single task).

The container for all OMERO applications is the `OmeroContext` (components/common/src/ome/system/OmeroContext.java). Based on the Spring configuration backing the context, it can be one of `client`, `internal`, or `managed`. The use of a ServiceFactory simplifies this usage for the client.

#### 20.13.1 Hibernate sessions

A Hibernate Session comprises a Unit-of-Work which translates for OMERO’s OME-Remote Objects model to a relational database. It keeps references to all Database-backed objects so that within a single session, object-identity stays constant and object changes can be persisted.

A session can span multiple calls by being disconnected from the underlying database transaction, and then reconnected to a new transaction on the next call (see components/server/src/ome/tools/hibernate/SessionHandler.java for the implementation).

For information about Events see [OMERO events and provenance](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/tools/hibernate/SessionHandler.java).
Omero Context Usage

Client context: user accesses a proxy acquired from a context instantiated with `OmeroContext.getclientContext()`. All objects are serialized and a transaction starts at the barrier to the Managed Context (see below).

Client context (with ServiceFactory): more commonly, user instantiates a new `ServiceFactory` and uses the public getters to acquire proxies.

Managed context: user accesses a wrapped service from a context instantiated with `OmeroContext.getManagedContext()`. Each call takes part in a single transaction.

Internal context: user accesses a raw service acquired from a context instantiated with `OmeroContext.getclientContext()`. No interception takes place. User must be careful to start transactions, open & flush sessions, and commit the transaction.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.
20.14 OMERO events and provenance

20.14.1 What is an event?

As described under *OmeroContext*, each method call takes place within a single application context (always the same), session, and event. Of these, only event is guaranteed to be unique for every task*. The components/server/src/ome/security/basic/EventHandler.java\(^{76}\) is responsible for creating new events.

20.14.2 Events as audit log

On each Database-update (INSERT/UPDATE/DELETE), an EventLog is created by a HibernateInterceptor which is then saved to the database at the end of the method call (in UpdateImpl).

20.14.3 Relationship to ModuleExecutions

The OMERO Event plays a similar role to the ModuleExecution in the OME 2 system. They both contain time of create/update/deletion, status, and type information. Event, however, has lost its ACL/permissions role. These values have been moved to embedded values represented by the Details object. Event also is not linked to all the created SemanticTypes as was ModuleExecution, and so cannot fully represent the provenance data needed by the AnalysisEngine. At such time as the AnalysisEngine is ported to Java, the ModuleExecution object will have to be added.

* Here we say “task” and not method call, because all method calls to a single stateful service instance belong to the same event. This is the nature of a stateful service. Logically, however, it is a single action.

See also:

Hibernate events\(^{77}\)

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

20.15 Properties

Under the etc/ directory in both the source and the binary distributions, several files are provided which help to configure OMERO.server:

etc/omero.properties\(^{78}\) Our central configuration file with all defaults

e tc/hibernate.properties\(^{79}\) Required by Hibernate since some properties are only configurable via a class-path:hibernate.properties file

e tc/logback.xml\(^{80}\) Logging configuration

e tc/build.properties\(^{81}\) The properties that you will most likely want to change

e tc/local.properties Local file overriding etc/build.properties (used by build only)

The most useful of the properties are listed in a glossary.

During the build, these files get stored in the blitz.jar and are read-only. On creation of an *OmeroContext*, the lookup for properties is (first wins):

- Properties passed into the constructor (if none, then the default properties in config.xml\(^{82}\))
- System.properties set via “java -Dproperty=value”
- Configuration files in order listed.

\(^{76}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/security/basic/EventHandler.java


\(^{82}\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/common/resources/ome/config.xml
This ordering is defined for the various components via “placeholder configurers” in:

- components/server/resources/ome/services/services.xml

Once configured at start, all values declared in one of the mentioned ways can be used in Spring configurations via the syntax:

```xml
<bean id="..." />
<property name="mySetter" value="${property.name}" />
</bean>
```

**Note:** This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

## 20.16 Using server queries internally

### Overview

This page is aimed at internal server developers and does not contain information for how to perform API queries. If that is the kind of information you are looking for, you may find [OMERO Application Programming Interface](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/resources/ome/services/services.xml) more useful as a starting point.

![Figure 20.3: omero.services.query](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/resources/ome/services/services.xml)

### 20.16.1 Introduction

The ome.services.queries package is intended to allow for the easy definition of queries by both developers and clients. Due to the fragility of HQL defined queries, a framework allowing for easy definition, multiple formats (Velocity templates, Database values, class files), and transparent lookup is critical.

---

83[https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/resources/ome/services/services.xml](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/resources/ome/services/services.xml)
Lookup happens among all QuerySources that are registered with the QueryFactory instance present in OMERO services. The first non-null Query instance returned by a QuerySource for a given String id is used.

Queries implement the HibernateCallback interface and are passed directly into an HibernateTemplate instance. Therefore, care should be taken as to which QuerySources are registered with the QueryFactory.

20.16.2 Parameters

Critical for using queries is the specification of named parameters, number of results to return, offset of the first result to return etc. These features are offered by the ome.parameters package. The ome.parameters.Parameters class is the starting point for building new parameters (although the ome.parameters.Filter object is used by some methods).

To specify parameters, instantiate a Parameters object either with or without a Filter object argument. The version with Filter object is useful for specifying the number of results to be returned and whether or not a java.util.Collection or a ome.model.IObject instance will be returned. For example,

```
Parameters p = new Parameters( new Filter().unique() );
```

will specify that the given query should return a single instance. An exception will be thrown if more than one result is found.

```
Parameters p = new Parameters( new Filter().unique().page(0,1) );
```

However, this will guarantee that only one result will be returned, since more than 1 result (“maxResults”) will be ignored. Here, an ordering of the results might make sense.

Once a Parameters instance is available, named parameters can be added using any of the `add…()` methods. These parameters will be dynamically bound during query preparation. For example, a query of the form:

```
select e from Experimenter e where omeName = :name
```

has one named parameter “name”, which can be specified by the call:

```
parameters.addString("name","<myNamHere>");
```

Positional parameters of the form

```
select e from Experimenter where omeName = ?
```

are not supported.

20.16.3 Adding queries

Subclassing query

Other than by defining String queries via `new QueryDef()` TBD, the easiest way to create queries is to subclass ome.services.query.Query. The only non-optional requirements on the Query implementor are then to define the (possibly optional) named parameters to the Query, and to override the “buildQuery” (which must call one and only one of “setQuery()” or “setCriteria()”)

Other than that, the Query implementor can enable filters on the Hibernate session (an attempt is made to clean up after the Query runs), and in general use any of the Hibernate session methods.
20.16.4 Defining a QuerySource

A more involved but perhaps more rewarding method would be to implement `QuerySource` and configure `QueryFactory` to lookup query ids also in your `QuerySource`. This would allow you to write Velocity (or Freemarker/Ruby/Python/Groovy…) `QuerySources` which use some form of templating or scripting to generate HQL queries.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

20.17 OMERO throttling

Throttling consists of reducing the total number of resources that one user or group can consume at a given time. The throttling service is a new component of `OMERO.blitz` which should ensure a more fair usage.

For example, each blitz server has a pre-defined maximum number of server threads. Any calls beyond this number must wait on a currently executing call to finish. Before throttling, a single user could consume all the available threads and all other users would have to wait.

With throttling, all invocations are placed on configurable on a queue which is worked on my any number of configurable slots. Each site can configure the number and type of slots based on which throttling strategy has been chosen.

20.17.1 Planning

Planned for milestone:3.0-Beta4, the infrastructure for throttling was committed to milestone 3.0-Beta3.1\textsuperscript{84} with the in-thread strategy, which uses the calling thread for execution. This provides the same semantics as the current blitz server.

Other strategies include:

- a per-session strategy
- a per-user strategy
- a per-group strategy

each of which allows the session, user, or group a fair slice of execution, but no more. Within each strategy, the order of operation is guaranteed not to change once the execution reaches the server. However, there is nothing the server can do to prevent re-ordering if two calls are made by the client simultaneously.

More advanced strategies are possible based on total consumed resources over some window, or even a service-level agreement (SLA) or Quality of Service (QoS)-style planning. All strategies must guarantee a proper method ordering.

It is also intended that the throttling service provide limits to memory usage, database hits within a single transaction, and total execution time.

20.17.2 Terminology

- **Slots** - are the number of available executions that a single session, user, or group can perform simultaneously on a single machine. (If the server is clustered, there will be the given number of slots per hosts)
- **Hard and soft limits** - hard limits throw an `OverUsageException` and require some form of compensation on the clients. Soft limits, on the other hand, simply slow down, or throttle, execution to give other operations a chance to succeed.
- **Strictness** - when a strategy is configured as strict, then once a session, user, or group has reached its limits, the hard or soft limit will be enforced even if no one else is using the server. A non-strict policy will “borrow” someone else’s slot for the duration of one execution.

See also:

Scaling Omero

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

\textsuperscript{84}https://trac.openmicroscopy.org/ome/milestone/3.0-Beta3.1
20.18 OMERO rendering engine

20.18.1 Description

The rendering component provides for the efficient rendering of raw pixels based on per-user display settings. A user can change settings and see them take effect in real time. Changes can also be persisted to the database and then viewed from another machine or even client.

20.18.2 Server-port

The rendering engine has been ported to also now sit on the server-side, though equally usable from any Java setting.

20.18.3 Optimizations

Here we have a listing of the various rendering engine optimizations that have taken place over time:

- Packed Integers (#449\textsuperscript{85})
- Region Based Rendering (#450\textsuperscript{86})
- Removal of RGB Rendering Model (#452\textsuperscript{87})

20.18.4 Compression

With r1744 and r1748, the rendering engine now supports compression. (#6\textsuperscript{88})

20.18.5 Design

The following diagrams describe the original design of the Rendering Engine. Designed initially for the client-side, much of this information needs to be updated. Textual explanations are included as notes in each diagram.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

20.19 Scaling Omero

There are several ways that OMERO, or any server system, can scale. Optimizing your system for more than one of these factors is non-trivial, but we try to lay out some guidelines below for what has worked, what almost certainly will not work, and what – under the right circumstances – might be optimal.

20.19.1 Concurrent invocations

The bottlenecks for concurrent invocations are:

- database connections
- server threads
- the router

\textsuperscript{85}https://trac.openmicroscopy.org/ome/ticket/449
\textsuperscript{86}https://trac.openmicroscopy.org/ome/ticket/450
\textsuperscript{87}https://trac.openmicroscopy.org/ome/ticket/452
\textsuperscript{88}https://trac.openmicroscopy.org/ome/ticket/6
Database connections

Database servers, in general, have a maximum number of allowed connections. In postgres, the default `max_connections` is 100, though in many cases this will be significantly lower due to the available shared memory (SHMMAX). If OMERO were to use direct connections to the database, after `max_connections` invocations, all further attempts to connect to the server would fail with “too many connection” exceptions. Instead, OMERO uses a connection pool in front of Postgres, which manages many more simultaneous attempts to connect to the database.

With the default `max_connection` set to 64, it is possible to execute 500 queries simultaneously without database exceptions. Instead, one receives server exceptions.

Server threads

In OMERO.blitz, too many (500+ on the default configuration) simultaneous invocations will result in ConnectionLost exceptions. We are currently working on ways to extend the number of single invocations on one server, but a simpler solution is to start another OMERO.blitz server.

20.19.2 Total throughput

The bottlenecks for throughput are:

- maximum message size
- server memory
- IO
The Container calls getInstanceOf during the initialization procedure. The singleton is created and the Container's registry is linked.

The Container calls activate() and the register() registers with the event bus. Now we're ready to accept incoming requests to load/render an image and to update rendering settings.

Events are processed as follows:
* LoadImage: A new rendering context is created and the rendering context proxy is linked to an ImageLoaded, which is posted on the event bus.
* RenderImage: The rendering context associated to the specified pixels set is asked to render the specified plane data. The rendered image is then linked to an ImageReRendered, which is posted on the event bus.
* SynchronizeImage: The .doUpdate method is invoked in order to trigger the appropriate call on a given server.

If the queue is empty, we wait for incoming events -- these are delivered by the event bus to eventFired(), which queues them. Because eventFired() runs in the same thread as the event bus, access to the queue is synchronized.

The Container calls terminate() and we exit. The Container will stop the event bus before asking us to terminate. This will prevent (if the singleton is in the Processing state) any events (fired during Processing) from being dispatched.

SqlAction

Internal server interface used to wrap all calls which speak JDBC directly. This allows special logic to be introduced where necessary for each RDBM.

Calls which use Hibernate for the cross-database conversion can use the org.hibernate.Session interface.

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.

See also:
OMERO.server and PostgreSQL Instructions about OMERO.server and PostgreSQL under UNIX & UNIX-like platforms.
OMERO.server and PostgreSQL Instructions about OMERO.server and PostgreSQL under Windows platforms.
OMERO.grid

#90689

Note: This documentation is for OMERO 5.2. This version is now in maintenance mode and will only be updated in the event of critical bugs or security concerns. OMERO 5.3 is expected before the end of 2016.
20.21 Model graph operations

Overview

When the OMERO server acts on its model objects it must determine the impact on related objects. For instance, deleting an image may entail deleting users’ rendering settings for that image, also the links to any datasets that the image is in.

Version 5.1.0 of the OMERO server included a new algorithm for determining which model objects to act on. Clients should be using the new 5.1 graph requests because the previous ones will be removed in OMERO 5.3. Understanding the details of the new implementation substantially assists in debugging or creating server operations that act on the directed graph of model objects.

20.21.1 Motivation

The OMERO model objects are interlinked. Plates may have wells whose samples come from multiple runs. Both datasets and well samples may have images, but in different ways. Datasets, wells, images, among others, may all be annotated. Images themselves are not simple: for example, they may be in a fileset, they may have ROIs drawn on them, they may share an instrument with the projection of that image. All these entities are separate objects that can be thought of as forming the nodes (vertices) of a directed graph of relationships.

Various operations supported by the OMERO server, most commonly moving objects to a different group, or deleting them, may implicitly include many related objects. For example, if one deletes a fileset, one also deletes the images from that fileset, and even the comments on those images. This section describes how the graph of model objects is traversed and how the target set of related objects is determined, under the new implementation of graph traversal first offered in version 5.1.0 of OMERO.

This technical detail is important to understand if one wishes to,

- adjust the set of related model objects that are included in operations
• change the types of OME-Remote Objects model objects or the permissible links among them

• fix bugs in the related request objects defined in Graphs.ice\textsuperscript{90} that may be submitted to OMERO sessions for execution.

20.21.2 Approach

Graph node states and transitions

In determining which model objects to process, and how, each corresponding graph node is in one of these states:

<table>
<thead>
<tr>
<th>adjective</th>
<th>rule format</th>
<th>Action enum</th>
<th>Orphan enum</th>
</tr>
</thead>
<tbody>
<tr>
<td>irrelevant</td>
<td>[E]{i}</td>
<td>EXCLUDE</td>
<td>IRRELEVANT</td>
</tr>
<tr>
<td>relevant</td>
<td>[E]{r}</td>
<td>EXCLUDE</td>
<td>RELEVANT</td>
</tr>
<tr>
<td>orphaned</td>
<td>[E]{o}</td>
<td>EXCLUDE</td>
<td>IS_LAST</td>
</tr>
<tr>
<td>attached</td>
<td>[E]{a}</td>
<td>EXCLUDE</td>
<td>IS_NOT_LAST</td>
</tr>
<tr>
<td>to delete</td>
<td>[D]</td>
<td>DELETE</td>
<td>n/a</td>
</tr>
<tr>
<td>to include</td>
<td>[I]</td>
<td>INCLUDE</td>
<td>n/a</td>
</tr>
<tr>
<td>outside</td>
<td>[O]</td>
<td>OUTSIDE</td>
<td>n/a</td>
</tr>
</tbody>
</table>

“enum” refers to the enumerations defined in GraphPolicy.java\textsuperscript{91}. Note also that, as for the introduction to Deleting in OMERO, “links” are simply edges in the graph, distinct from the classes implementing ILink.java\textsuperscript{92} which themselves have several links, not least to their parent and child objects.

\textsuperscript{90}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/resources/omero/cmd/Graphs.ice
\textsuperscript{91}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/services/graphs/GraphPolicy.java
\textsuperscript{92}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/model/src/ome/model/ILink.java
After quantization, we can apply a series of transformations that operate on the codomain (a sub-interval \([0, 255]\)) of the initial quantum map to enhance the image (these are referred to as enhancements in the spatial domain). Here are the key classes that are involved in this process:

**CodomainMap**
- `setContext(void)`
- `getContext(void)`
- `transform(void)`
- `transform(int)`
- `buildLUT()`
- `addValueMap(int)`
- `removeValueMap()`
- `mapValue(int)`
- `mapValue(int, int)`

**CodomainMapContext**
- `setContext()`
- `getContext()`
- `transformContext()`
- `buildContext()`
- `setCodomainMap()`
- `copy()`
- `equals()`

**Quantization**
- `setQuantizationContext()`
- `setQuantization()`
- `getQuantization()`
- `getQuantizationContext()`
- `quantize()`
- `dequantize()`

**RenderingDef**
- `setModel()`
- `getModel()`
- `setDef()`
- `getDef()`
- `setWaveRendering()`
- `setSpatialRendering()`
- `setSpatialDomainEntertainment()`

**ChannelBindings**
- `setDef()`
- `getDef()`
- `setChannelMapping()`
- `getChannelMapping()`

Queues the contexts that define the spatial domain transformations that have to be applied to the image. A lookup table is built by mapping all maps (in the same order as their contexts were sequenced) into a single transformation and then by applying this map to each value in the interval [0, 255] of the codomain interval (note that, in order to compose the maps, this interval has to be both the domain and codomain of each transformation.)

The LUT is re-built every time the definition of the codomain interval or the state of the queue changes. Contexts are prioritized by kind (add and update make copies) because we want to exclude the possibility that a context's state can be modified after the lookup table is built.
When traversal begins, the target objects are to be included (e.g., for Chgr2\(^93\)) or deleted (e.g., for Delete2\(^94\)) and other objects are irrelevant.

A list of transition rules is associated with the requested operation. Each of the target objects is examined in turn and the rules matched against the state of that object and of those directly linked to it in either direction. If a rule matches, it may either abort the operation with an error condition or, more usually, change the state of any of the objects it matches. Changed objects are themselves queued for examination and rule matching. The traversal is complete when all queued objects have been examined with no further transition rule matches. Rules that can abort the operation are checked only after the other rules have completed processing. GraphTraversal.java\(^95\)'s planOperation method is at the heart of this matching process.

### Further graph node states

Usual behavior is for orphaned objects related to the target objects to be included in the operation, but not the otherwise-attached objects, the non-orphans who have excluded parents that are to be neither deleted nor included. The related children that may be orphans are exactly those identified as being relevant. Transition rules match these against excluded parents to discover if the relevant objects do have any qualifying parents, changing them to be attached objects. If no further rules match and some objects remain as relevant, then they are automatically changed to orphans and examined for further rule matches. After that processing completes, attached objects are changed back to being relevant to confirm that excluded qualifying parents still exist to change them to being attached: this is necessary in case, after an object was considered attached, other rules changed all those qualifying parents from being excluded so that the object is now an orphan.

---

\(^93\)http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Chgrp2.html

\(^94\)http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Delete2.html

\(^95\)https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/services/graphs/GraphTraversal.java
Objects that are changed to be outside are effectively rendered invisible, outside consideration in the execution phase. In the execution of an operation the graph traversal code removes links between included and excluded objects, but it allows links to remain between outside objects and other objects. Outside objects typically implement IGlobal.java and have no owner or group.

An additional aspect of objects’ state is if permissions are to be checked for them. For instance, typically I may move only my own objects to a different group, but if another user tags my image with my tag, then I may still move my image and tag to a different group, also moving that link even though it is not my own object: in that case, permissions checking is disabled for that ImageAnnotationLink. All objects initially have permissions checking enabled, but the consequence of a rule may be to disable permissions checking, and if an object with permissions checking disabled matches a further rule, the objects changed by that rule also have permissions checking disabled.

20.21.3 Configuration

Defining the model graph transition rules

To reduce its complexity, GraphTraversal.java does not include specific detail of how to traverse the graph of OME-Remote Objects model objects. Instead, subclasses of GraphPolicy.java guide the traversal of the model object graph, configured by blitz-graph-rules.xml which names and defines the lists of transition rules. The named lists of rules are associated with request object classes by the definition of the graphRequestFactory bean in blitz-servantDefinitions.xml, which also specifies which model object properties may never be set to null in executing any requested operation.

blitz-graph-rules.xml begins with a comment that provides a key to the notation used for transition rules. The rules name and match model objects based on the state of the graph nodes, the types of the corresponding objects, the permissions the user has on those objects, and the names of the properties linking the objects. To illustrate this, the following sections briefly describe some different kinds of rule from the deleteRules list.

Propagating deletion

```plaintext
p:matches="L:ILink.parent = [D], L.child = C:[E]{o}/d"
p:changes="C:[D]"
```

---

96 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/model/src/ome/model/IGlobal.java
97 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/services/graphs/GraphTraversal.java
98 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/services/graphs/GraphPolicy.java
99 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/resources/ome/services/blitz-graph-rules.xml
100 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/resources/ome/services/blitz-servantDefinitions.xml
101 https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/resources/ome/services/blitz-graph-rules.xml
If an ILink’s parent (e.g., a dataset) is to be deleted, and its child (e.g., an image) is orphaned and deletable by the user, then delete the child also.

```
p:matches="PlateAcquisition[D].wellSample = WS:WellSample[E]"
p:changes="WS:[D]"
```

If a plate acquisition (run) is to be deleted, also delete its well samples (fields).

```
p:matches="Fileset[D] = I:Image[E].fileset"
p:changes="I:[D]"
```

If a fileset is to be deleted, then also delete its images.

```
p:matches="T:Thumbnail[E].pixels =/!o [D]"
p:changes="T:[D]/n"
```

If the pixels of a thumbnail are to be deleted, and are owned by a different user, then delete the thumbnail regardless of permissions on it.

**Curtailing deletion**

```
p:matches="Well[D].plate = C:[E]{!a}" p:changes="C:{a}"  
```

If a well is to be deleted but its plate is excluded and not attached, regard the plate as attached.

```
p:matches="C:Channel[E]{r}.pixels = Pixels[E]{i}"
p:changes="C:{a}"  
```

If an irrelevant pixels object has a relevant channel, then regard the channel as attached.

```
p:matches="Pixels[D].relatedTo = P:[E]{!a}" p:changes="P:{a}"  
```

If a pixels object is to be deleted, regard any related, excluded pixels objects as attached. Because the pixels of an image are related to the pixels of a projection of that image, this rule prevents the deletion of an image from causing inadvertent deletion of the image’s projections.

```
p:matches="L:ILink[!D].parent = [E]/d, L.child = C:[E]{r}"
p:changes="C:{a}"  
```

If an ILink that is not to be deleted itself has a deletable, excluded parent and a relevant child, regard that child as attached.

**Other kinds of transition rule**

```
p:matches="E:IEnum[E]"
p:changes="E:[O]"
```

Regard excluded IEnum objects as being outside the operation. (Rules do not need to match on links among multiple objects.)
p:matches="F:Fileset[D].images = [D], F.images = ![D]"
p:error="may not split {F}"

Throw an error if a fileset that is not to be deleted includes an image that is to be deleted and an image that is not to be deleted.

In reviewing the chgrpRules list, one sees conditions that require matching $to_private or !$to_private. A request, in this case Chgrp2I.java, may set arbitrary conditions upon which rules may be predicated. The to_private condition, or its absence, is used to cause different behavior when the objects are being moved into a private group.

### 20.21.4 Logging

**Changing the log level**

It is informative to observe the sequence of rule applications as the graph is traversed and decisions about model objects are made. To do so requires configuring Omero logging for the server, specifically etc/logback.xml. To activate graph traversal debug logging, adjust the ends of the lines,

```xml
<logger name="omero.cmd.graphs" level="INFO"/>
<logger name="ome.services.graphs" level="INFO"/>
```

such that they instead read,

```xml
<logger name="omero.cmd.graphs" level="DEBUG"/>
<logger name="ome.services.graphs" level="DEBUG"/>
```

The resulting extra information in var/log/Blitz-0.log is of particular assistance in debugging: it pinpoints the rule applications that caused incorrect determinations of what action to take with model objects. Note that a * suffix on a model object referenced in the logs indicates that permissions are not to be checked for it.

**Expanding the reports of transition rule matches**

In the previous section, it can be seen that model objects that match rule conditions may be named. For example, in,

```xml
p:matches="Fileset[D] = I:Image[E].fileset"
p:changes="I:[D]"
```

the image is named I. When a rule matches, the debug logging reports which model object matched each name. If it remains unclear why a rule matched, further objects may be named. For example, changing the first line to name the fileset,

```xml
p:matches="F:Fileset[D] = I:Image[E].fileset"
```

would also report in the log which fileset matched the rule.

### 20.21.5 Encouragement

On first reading, the above may feel daunting. If model object graph traversal is not working as desired, thus requires adjustment, review of debug logs from var/log/Blitz-0.log typically pinpoints the cause and a minor adjustment to blitz-graph-rules.xml often suffices as the fix, with integration tests providing reassurance that the adjustment was acceptable. Sometimes it can take time and thought to devise that fix, but one can expect small changes to suffice to fix most bugs. In getting this new graph

---


103 [https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/etc/logback.xml](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/etc/logback.xml)

104 [https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/resources/ome/services/blitz-graph-rules.xml](https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/resources/ome/services/blitz-graph-rules.xml)
traversal implementation to initially pass integration testing, no test failures required a substantial rethink of the basic approach and GraphTraversal.java\textsuperscript{105} itself did not require a significant rewrite.

The actual lists of transition rules arose in part as a way to achieve the desired behavior and are not yet as simple and comprehensive as they could be. While they necessarily reflect the inherent complexity of the object model of OME-

Remote Objects, there is potential for reviewing the rule lists and, perhaps with some additional marker interfaces, making them more succinct and regular. Incremental movements toward this goal are worth pursuing.

### 20.21.6 Options

Every one of the request object classes introduced in the new implementation of graph traversal is a derived class of GraphModify\textsuperscript{2106} and inherits data members that configure its operation. Each request may define additional data members for options specific to it, for instance Chgrp\textsuperscript{2107} requires the ID of the target group to be specified. The data members offered by all of the new requests are,

- **targetObjects** specifies which model objects the operation is to target
- **childOptions** specifies types of model objects (and, for annotations, namespaces) that should always or never be included in the operation (i.e. always considered to be orphans, or attached, regardless of excluded parents)
- **dryRun** specifies if the request is to determine which model objects would be included in and deleted by the operation, without actually executing the operation.

### 20.21.7 SkipHead

The request object classes derived from GraphModify\textsuperscript{108} allow specification of the target objects with reference to a common parent. For example, setting the request’s type to /Plate/Well/WellSample/Image targets the images of a specific plate.

The SkipHead\textsuperscript{109} request offers a similar feature. It wraps an inner request data member that starts acting only after graph traversal reaches types listed in startFrom. For that inner request to behave as if given the previous /Plate/Well/WellSample/Image type, a plate may be given in targetObjects and Image named in startFrom.

This feature is achieved by running the initial request with dryRun set to true and the graph traversal policy modified so as not to examine included nodes of types listed in startFrom. A subsequent request then runs, targeting the startFrom model objects that were included in the first request.

### 20.21.8 Compatibility

The previous implementation of model graph traversal has request object classes derived from GraphModify\textsuperscript{110} instead of the newer GraphModify2\textsuperscript{111}. For API compatibility a set of “facade” requests are defined by the new implementation that offer an approximation of the older API but run the newer code under the hood. For instance, Chgrp\textsuperscript{112} is implemented by ChgrpFacade\textsuperscript{113} by means of Chgrp2I.java\textsuperscript{114}.

\textsuperscript{105}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/server/src/ome/services/graphs/GraphTraversal.java

\textsuperscript{106}http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/GraphModify2.html

\textsuperscript{107}http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Chgrp2.html

\textsuperscript{108}http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/GraphModify.html

\textsuperscript{109}http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/SkipHead.html

\textsuperscript{110}http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/GraphModify.html

\textsuperscript{111}http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/GraphModify2.html

\textsuperscript{112}http://downloads.openmicroscopy.org/latest/omero/api/slice2html/omero/cmd/Chgrp.html

\textsuperscript{113}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/src/omero/cmd/graphs/ChgrpFacadeI.java

\textsuperscript{114}https://github.com/openmicroscopy/openmicroscopy/blob/v5.2.6/components/blitz/src/omero/cmd/graphs/Chgrp2I.java
INDEX

Symbols

–advanced-help
  omero-import command line option, 9
–all
  omero-help command line option, 8
–archived
  omero-fs-images command line option, 182
–check
  omero-fs-sets command line option, 181
–debug DEBUG
  omero-import command line option, 10
–depth DEPTH
  omero-import command line option, 10
–dry-run
  command line option, 20, 23
–email EMAIL
  omero-import command line option, 10
–exclude
  command line option, 19, 21
–extended
  omero-fs-images command line option, 182
  omero-fs-sets command line option, 181
–force-rewrite
  omero-admin-start command line option, 176
  omero-admin-stop command line option, 176
–foreground
  omero-admin-start command line option, 176
–groups
  omero-fs-usage command line option, 184
–include
  command line option, 19, 21
–java-help
  omero-import command line option, 9
–limit <LIMIT>
  omero-fs-images command line option, 182
  omero-fs-sets command line option, 181
–list
  omero-help command line option, 8
–logs
  omero-import command line option, 11
–managed
  omero-fs-repos command line option, 180
–markers
  setup.py-test command line option, 304
–no-move
  omero-fs-rename command line option, 183
–offset <OFFSET>
  omero-fs-images command line option, 182
  omero-fs-sets command line option, 181
–order <[newest,oldest, prefix]>
  omero-fs-images command line option, 182
  omero-fs-sets command line option, 181
–ordered
  command line option, 20, 22
–recursive
  omero-help command line option, 8
–repeat <number>
  py.test command line option, 304
–report
  command line option, 20, 22
  omero-fs-usage command line option, 184
  omero-import command line option, 10
–size_only
  omero-fs-usage command line option, 184
–skip SKIP
  omero-fs-usage command line option, 184
–sudo ADMINUSER
  omero-login command line option, 15
–units {K,M,G,T,P}
  omero-fs-usage command line option, 184
–upload
  omero-import command line option, 11
–wait WAIT
  omero-fs-usage command line option, 184
–with-transfer <WITH_TRANSFER [WITH_TRANSFER ...]>
  omero-fs-sets command line option, 181
–without-images
  omero-fs-sets command line option, 181
-T, –target TARGET
  omero-import command line option, 10
-f
  omero-import command line option, 10
-g GROUP, –group GROUP
  omero-login command line option, 15
-h, –help
  omero-admin-start command line option, 176
  omero-admin-stop command line option, 176
  omero-fs-images command line option, 182
  omero-fs-rename command line option, 182
  omero-fs-repos command line option, 180
  omero-fs-sets command line option, 181
A
Action, 84
addData() (omero.grid.Table method), 374
Administrator, 81
AgentEvent, 433
AgentEventListener, 433
Annotate, 84
B
broken, 306
C
ch.qos.logback.core.Appender.error, 226
CMAKE_INCLUDE_PATH, 366
CMAKE_LIBRARY_PATH, 366
columns (omero.grid.Data attribute), 373
command
omerof-so usage command line option, 183
omerof-import command line option, 9
py.test command line option, 304
setup.py test command line option, 304
-k KEY, --key KEY
omerof-login command line option, 15
-k <string>
setup.py test command line option, 303
-m <marker>
setup.py test command line option, 303
p PORT, --port PORT
omerof-login command line option, 15
-s
setup.py test command line option, 304
-s SERVER, --server SERVER
omerof-login command line option, 15
-t <test_path>, --test-path <test_path>
setup.py test command line option, 303
-u USER, --user USER
omerof-login command line option, 15
-w PASSWORD, --password PASSWORD
omerof-login command line option, 15
D
Delete, 84
DYLD_LIBRARY_PATH, 89, 369
E
Edit, 84
environment variable
Index 650
EventBus, 433
EventBusListener, 433
G
getAllMetadata() (omero.grid.Table method), 374
getHeaders() (omero.grid.Table method), 373
getMetadata() (omero.grid.Table method), 374
getNumberOfRows() (omero.grid.Table method), 373
getWhereList() (omero.grid.Table method), 374
Group member, 81
Group owner, 81
GTEST_ROOT, 365
I
Ice.Ipv6, 241
ICE_CONFIG, 301, 302, 307, 349
ICE_HOME, 364, 366
initialize() (omero.grid.Table method), 374
intermittent, 306
J
JAVA_OPTS, 228
jvm.fileDescriptorCountRatio, 226
L
lastModification (omero.grid.Data attribute), 373
LD_LIBRARY_PATH, 89, 369
long_running, 306
M
ManualStrategy, 223
Mix data, 84
Move between groups, 84
O
ome.io.nio.PixelsService.minmaxTimes, 226
ome.io.nio.PixelsService.tileTimes, 226
ome.services.eventlogs.EventLogQueue.priorityCount, 226
omerof admin, 173, 232
OMERO Documentation, Release 5.2.6

Index 651

- cleane, 495
deploy, 198, 232
diagnostics, 192, 199, 224
e-mail, 202
jvmcfg, 192
start, 171, 195, 197–199, 201, 232
stop, 171, 195, 198
omero chgrp, 18
omero config, 91, 173, 174
def, 174, 175
drop, 175
edit, 235
get, 174, 194
load, 175
parse, 91
set, 171, 174, 175, 195, 229, 235, 239, 269, 408
omero db, 173
script, 173, 510
omero delete, 18, 20
omero fs, 180
images, 181
rename, 182
repos, 180, 194
sets, 180
usage, 183
omero group, 177
add, 178, 179
adduser, 179
copyusers, 179
list, 179
removeuser, 179
omero help, 334
omero hql, 299, 521
omero ldap
create, 177, 218
setdn, 177
omero node, 196
NAME
stop, 198
omero-slave
start, 199
omero obj, 203, 333, 334
new, 333
update, 334
omero script, 383
omero tag, 17
create, 17
createrset, 17
link, 18
list, 17
listsets, 18
omero user, 177
add, 177
joingroup, 179
leavegroup, 179
list, 179
omero web
config, 188
omero-admin-start command line option
- h, -help, 176
omero-admin-stop command line option
- force rewrite, 176
- h, -help, 176
omero-fs-images command line option
- archived, 182
- extended, 182
- limit <LIMIT>, 182
- offset <OFFSET>, 182
- order <{newest,oldest,prefix}>, 182
- style {plain,csv,sql}, 182
- h, -help, 182
omero-fs-rename command line option
- no-move, 183
- h, -help, 182
omero-fs-repos command line option
- managed, 180
- style <{plain,csv,sql}>, 180
- h, -help, 180
omero-fs-sets command line option
- check, 181
- extended, 181
- limit <LIMIT>, 181
- offset <OFFSET>, 181
- order <{newest,oldest,prefix}>, 181
- style <{plain,csv,sql}>, 181
- with-transfer <WITH_TRANSFER
  [WITH_TRANSFER ...]>], 181
- without-images, 181
- h, -help, 181
omero-fs-usage command line option
- groups, 184
- report, 184
- size only, 184
- style {plain,csv,sql}, 183
- units {K,M,G,T,P}, 184
- wait WAIT, 184
- h, -help, 183
omero-help command line option
- all, 8
- list, 8
- recursive, 8
- command, 8
omero-import command line option
- advanced-help, 9
- debug DEBUG, 10
- depth DEPTH, 10
- email EMAIL, 10
- java-help, 9
- logs, 11
- report, 10
- skip SKIP, 10
- upload, 11
- T, -target TARGET, 10
- f, 10
- h, -help, 9
omero-login command line option
- sudo ADMINUSER, 15
- g GROUP, -group GROUP, 15
- k KEY, -key KEY, 15
- p PORT, -port PORT, 15
ResponseEvent, 435
rowNumbers (omero.grid.Data attribute), 373

S
setAllMetadata() (omero.grid.Table method), 374
setMetadata() (omero.grid.Table method), 374
setup.py-test command line option
  –markers, 304
  -h, –help, 304
  -k <string>, 303
  -m <marker>, 303
  -s, 304
  -t <test_path>, --test-path <test_path>, 303
slice() (omero.grid.Table method), 373
SLICEPATH, 129, 290, 366
StateChangeEvent, 435

U
update() (omero.grid.Table method), 374

V
values (omero.grid.DoubleColumn attribute), 372
VERBOSE, 366
View, 84

W
webclient, 396
webgateway, 396