



CLC Science Server

User manual

Administrator Manual for
CLC Science Server 2.2 including CLC Bioinformatics Database
Windows, Mac OS X and Linux

May 24, 2013

This software is for research purposes only.

CLC bio
Finlandsgade 10-12
DK-8200 Aarhus N
Denmark



Contents

1	Introduction	8
1.1	System requirements	8
1.2	Licensing	9
2	Installation	11
2.1	Quick installation guide	11
2.2	Installing the database	11
2.2.1	Download and install a Database Management System	11
2.2.2	Create a new database and user/role	12
2.2.3	Initialize the database	12
2.3	Installing and running the Server	13
2.3.1	Installing the Server software	14
2.4	Silent installation	15
2.5	Upgrading an existing installation	16
2.5.1	Upgrading major versions	16
2.6	Allowing access through your firewall	16
2.7	Downloading a license	17
2.7.1	Windows license download	17
2.7.2	Mac OS license download	17
2.7.3	Linux license download	18
2.8	Starting and stopping the server	18
2.8.1	Microsoft Windows	18
2.8.2	Mac OS X	18
2.8.3	Linux	19
2.9	Installing relevant plug-ins in the Workbench	20

3	Configuring and administering the server	21
3.1	Logging into the administrative interface	21
3.2	Adding locations for saving data	21
3.2.1	Adding a database location	21
3.2.2	Adding a file system location	22
	Important points about the CLC Server data in the file system locations . .	23
	File locations for job node set-ups	23
3.2.3	Rebuilding the index	23
3.3	Accessing files on, and writing to, areas of the server filesystem	24
3.4	Changing the listening port	25
3.5	Changing the tmp directory	26
3.5.1	Job node setups	26
3.6	Setting the amount of memory available for the JVM	26
3.7	Limiting the number of cpus available for use	27
3.8	Other configurations	27
3.8.1	HTTP settings	27
3.8.2	Audit log settings	27
3.8.3	Deployment of server information to CLC Workbenches	28
3.9	Server plug-ins	28
3.10	Queue	28
4	Managing users and groups	30
4.1	Logging in the first time - root password	30
4.2	User authentication using the web interface	30
4.2.1	Managing users using the web interface	31
4.2.2	Managing groups using the web interface	32
4.3	User authentication using the Workbench	33
4.3.1	Managing users through the Workbench	33
4.3.2	Managing groups through the Workbench	33
4.3.3	Adding users to a group	34
4.4	User statistics	34
5	Access privileges and permissions	36

5.1	Controlling access to data	36
5.1.1	Setting permissions on a folder	37
	Permissions on the recycle bin	38
5.1.2	Technical notes about permissions and security	38
5.2	Global permissions	39
6	Job Distribution	40
6.1	Model I: Master server with execution nodes	41
6.1.1	Master and job nodes explained	41
6.1.2	User credentials on a master-job node setup	41
6.1.3	Configuring your setup	42
6.1.4	Installing Server plug-ins	43
6.2	Model II: Master server submitting to grid nodes	44
6.2.1	Requirements for CLC Grid Integration	44
	Supported job submission systems	44
	DRMAA for PBS Pro	45
6.2.2	Technical overview	45
6.2.3	Setting up the grid integration	46
6.2.4	Licensing of grid workers	47
6.2.5	Configuring licenses as a consumable resource	48
6.2.6	Configure grid presets	48
6.2.7	Controlling the number of cores utilized	50
6.2.8	Other grid worker options	51
6.2.9	Testing a Grid Preset	52
6.2.10	Client-side: starting CLC jobs on the grid	52
	Installing the CLC Workbench Client Plug-in	52
	Starting grid jobs	52
6.2.11	Grid Integration Tips	52
6.2.12	Understanding memory settings	54
7	External applications	55
7.1	External applications integration: Basic configuration	56
7.2	External applications integration: Post-processing	58

7.3	External applications integration: Stream handling	58
7.4	External applications integration: Environment	59
7.4.1	Environmental Variables	59
7.4.2	Working directory	59
7.4.3	Execute as master process	59
7.5	External applications integration: Parameter flow	59
7.6	External applications integration: Velvet	60
7.6.1	Installing Velvet	60
7.6.2	Running Velvet from the Workbench	61
7.6.3	Understanding the Velvet configuration	62
7.7	Troubleshooting	63
7.7.1	Checking the configuration	63
7.7.2	Check your third party application	63
7.7.3	Is your Import/Export directory configured?	63
7.7.4	Check your naming	63
8	Workflows	65
8.1	Installing and configuring workflows	65
8.2	Executing workflows	65
9	Appendix	68
9.1	Troubleshooting	68
9.1.1	Check set-up	68
9.1.2	Bug reporting	68
9.2	Database configurations	70
9.2.1	Configurations for MySQL	70
9.3	SSL and encryption	70
9.3.1	Enabling SSL on the server	70
	Creating a PKCS12 keystore file	71
9.3.2	Logging in using SSL from the Workbench	71
9.3.3	Enabling redirection from non-ssl port to ssl-enabled port	71
9.3.4	Logging in using SSL from the <i>CLC Server Command Line Tools</i>	72

Bibliography	74
Index	74

Chapter 1

Introduction

The *CLC Server* is the central part of CLC bio's enterprise solutions. You can see an overview of the server solution in figure 1.1.

For documentation on the customization and integration, please see the developer kit for the server at <http://www.clcdeveloper.com>.

The *CLC Science Server* is shipped with the following tool that can be started from the Workbench:

- BLAST (creation of databases and running BLAST)
- Find primer binding sites
- Annotate secondary peaks (Sanger data)
- External applications (integrating with 3rd party programs on the server)

The functionality of the *CLC Science Server* can be extended by installation of Server plug-ins. The available plug-ins can be found at http://www.clcbio.com/server_plugins.

1.1 System requirements

The system requirements of *CLC Science Server* are:

- Windows XP, Windows Vista, or Windows 7, Windows Server 2003 or Windows Server 2008
- Mac OS X 10.6 or later. However, Mac OS X 10.5.8 is supported on 64-bit Intel systems.
- Linux: Red Hat or SUSE
- 32 or 64-bit
- 256 MB RAM required

Minimum 2GB of RAM recommended, depending on the type of analysis to be performed.

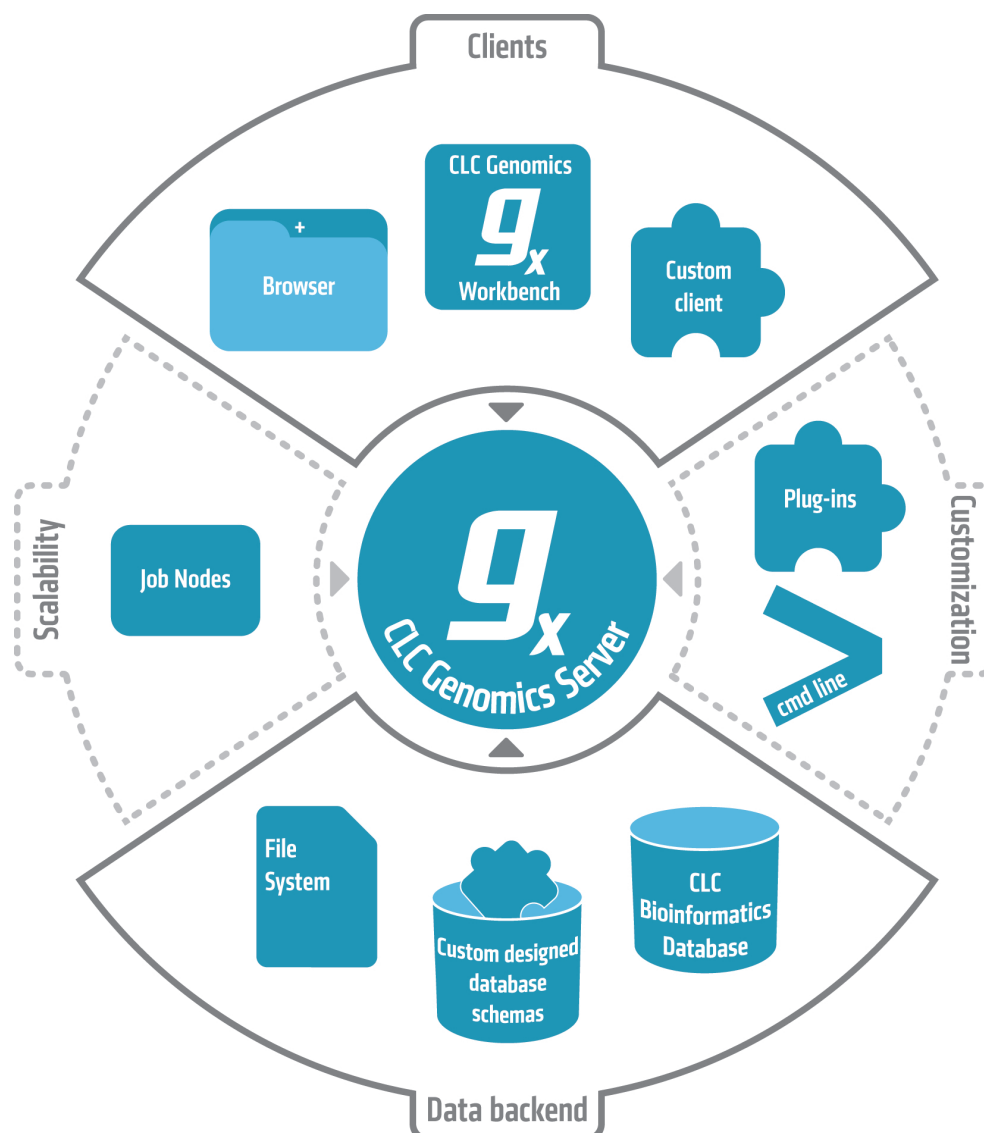


Figure 1.1: An overview of the server solution from CLC bio. Note that not all features are included with all license models.

1.2 Licensing

There are three kinds of licenses needed for running the CLC Science Server:

- A license for the server. This is needed for running the server. The license will allow a certain number of sessions (i.e. number of log-ins from e.g. the web interface or the workbench). The number of sessions is part of the agreement with CLC bio when you purchase a license.
- A special license if you are running the server on a grid system (this is explained in detail in section 6.2.4)
- A license for the workbench. The Workbench is needed to start analyses on the server and view the results. Find the user manuals and deployment manual for the Workbenches at <http://www.clcbio.com/usermanuals>

The following chapter on installation will give you more information about how to obtain and deploy the license for the server.

Chapter 2

Installation

2.1 Quick installation guide

The following describes briefly the steps needed to set up a *CLC Science Server 2.2 including CLC Bioinformatics Database* with pointers to more detailed explanation of each step. If you are going to set up execution nodes as well, please read section 6 first.

1. Download and run the server installer. As part of the installation, choose to start the server (section 2.3).
2. Run the license download script that is part of the installation (section 2.7).
3. The script will automatically download a license file and place it in the server installation directory under `licenses`.
4. Restart the server (section 2.8).
5. Log into the server using a web browser (note that the default port is 7777) with username **root** and password **default** (section 3).
6. Change the root password (section 4.1).
7. Configure authentication mechanism and optionally set up users and groups (section 4.2).
8. Add data locations (section 3.2).
9. Download and install plug-ins *in the Workbench* needed for the Workbench to contact the server (section 2.9).
10. Check your server set-up using the **Check set-up** link in the upper right corner as described in section 9.1.1.
11. Your server is now ready for use.

2.2 Installing the database

2.2.1 Download and install a Database Management System

If you do not already have an existing installation of a Database Management System (*DBMS*) you will have to download and install one. The CLC Bioinformatics Database can be used with a

number of different DBMS implementation. Choosing the right one for you and your organization depends on many factors such as price, performance, scalability, security, platform-support, etc.

Information about the supported solutions are available on the links below.

- MySQL: <http://dev.mysql.com/downloads/>
- PostgreSQL: <http://www.postgresql.org/>
- Microsoft SQL Server: <http://www.microsoft.com/SQL/>
- Oracle: <http://www.oracle.com/>

If you have further questions about the installation of the DBMS, please contact support@clcbio.com. See section 9.2 for guidance on special configurations for the DBMS.

2.2.2 Create a new database and user/role

Once your DBMS is installed and running you will need to create a database for containing your CLC data. We also recommend that you create a special database-user (sometimes called a database-role) for accessing this database.

Consult the documentation of your DBMS for information about creating databases and managing users/roles.

2.2.3 Initialize the database

Before you can connect to your database from a CLC Workbench or Server it must be initialized. The initialization creates the required tables for holding objects, and prepares an index used for searching. Initialization is performed with the CLC Bioinformatics Database Tool (figure 2.1).

- Install the CLC Bioinformatics Database Tool on a client machine, and start the program.
- Fill in the fields with the required information.
 - Hostname: The fully-qualified hostname of the server running the database.
NOTE: The same hostname must be used every time you connect to the database
 - Port: The TCP/IP listening port on the database server
 - Database name: The name of the database you created in the previous section
 - Username: the name of the user/role you created in the previous section
 - Password: the password for the user/role.
- To re-initializing an existing CLC database you must check the "Delete Existing..." checkbox.
NOTE: ANY DATA ALREADY STORED IN THE CLC DATABASE WILL BE DELETED.
- Click the Initialize Database button to start the process.

While the program is working the progress-bar will show the status and the transcript will show a log of actions, events and problems. If anything goes wrong, please consult the transcript for

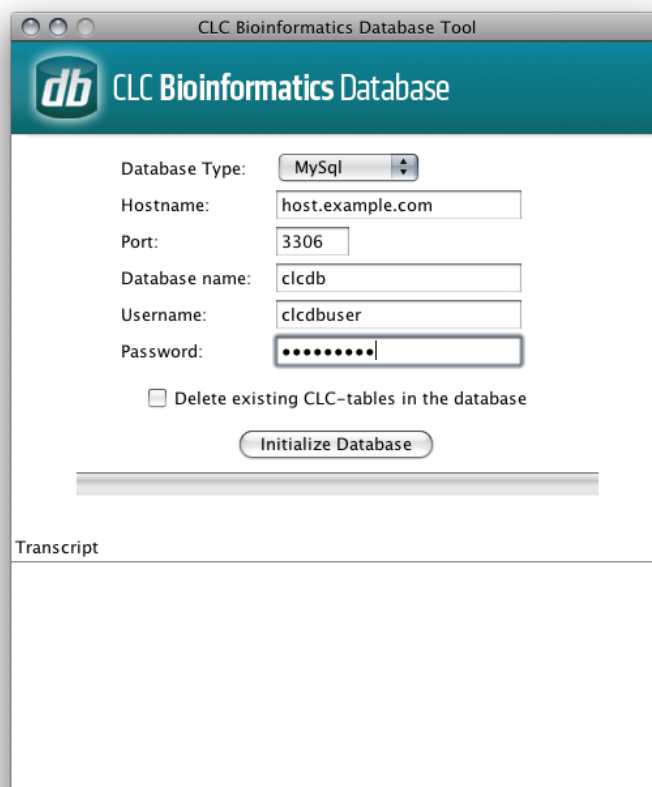


Figure 2.1: The CLC Bioinformatics Database tool

more information. If you need assistance, please contact support@clcbio.com, and include the contents of transcript.

If the initialization is successful, the status bar will display this message: *Database successfully initialized*. You can now close the CLC Bioinformatics Database Tool.

2.3 Installing and running the Server

Getting the *CLC Science Server* software installed and running involves, at minimum, these steps:

1. Install the software.
2. Ensure the necessary port in the firewall is open.
3. Download a license.
4. Start the Server and/or configure it as a service.

All these steps are covered in this section of the manual. Further configuration information, including for job nodes, grid nodes, and External Applications, are provided in later chapters.

Installing and running the *CLC Science Server* is straightforward. However, if you do run into troubles, please refer to the troubleshooting section in Appendix 9.1, which provides tips on how to troubleshoot problems yourself, as well as how to get help.

2.3.1 Installing the Server software

The installation can only be performed by a user with administrative privileges. On some operating systems, you can double click on the installer file icon to begin installation. Depending on your operating system you may be prompted for your password (as shown in figure 2.2) or asked to allow the installation to be performed.

- On Windows 7 or Vista, you will need to right click on the installer file icon, and choose to **Run as administrator**.
- For the Linux-based installation script, you would normally wish to install to a central location, which will involve running the installation script as an administrative user - either by logging in as one, or by prefacing the command with `sudo`. Please check that the installation script has executable permissions before trying to execute it.



Figure 2.2: Enter your password.

Next you will be asked where to install the server 2.3. If you do not have a particular reason to change this, simply leave it at the default setting. The chosen directory will be referred to as the *server installation directory* throughout the rest of this manual.

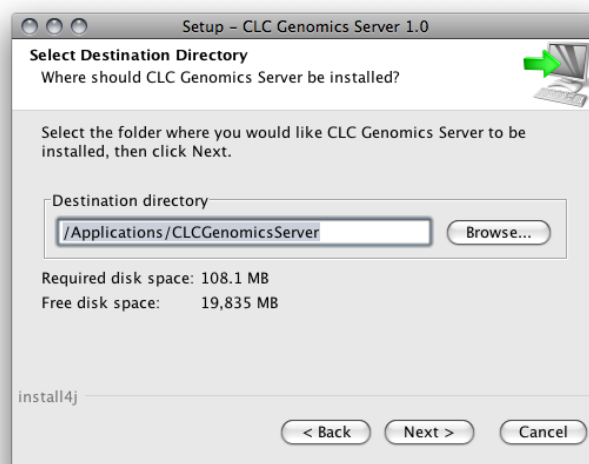


Figure 2.3: Choose where to install the server.

The installer allows you to specify the maximum amount of memory the CLC Server will be able to utilize 2.4. The range of choice depends on the amount of memory installed on your system and on the type of machine used. On 32 bit machines you will not be able to utilize more than 2 GB of memory – on 64-bit machines there is no such limit.

If you do not have a reason to change this value you should simply leave it at the default setting.

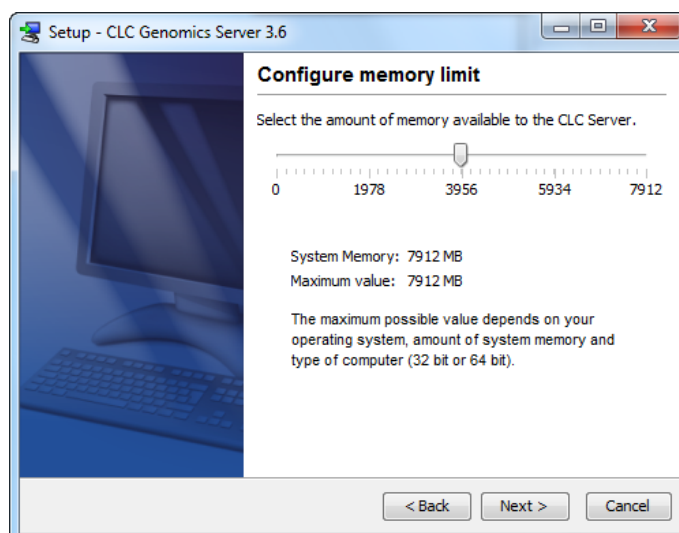


Figure 2.4: Choose the maximum amount of memory used by the server.

If you are installing the server on a Windows system you will be able to choose if the service is started manually or automatically by the system.

The installer will now extract the necessary files.

On a Windows system, if you have chosen that the service should be started automatically, the service should also start running at this point. Please note that if you do not already have a license file installed, then the *CLC Science Server* process will be running in a limited capacity at this point. Downloading a license is described in section 2.7.

Information on stopping and starting the *CLC Science Server* service is provided in section 2.8.

2.4 Silent installation

The installer also has a silent installation mode which is activated by the `-q` parameter when running the installer from a command line, e.g.

```
CLCGenomicsServer_3_0.exe -q
```

On Windows, if you wish to have console output, `-console` can be appended as the *second parameter* (this is only needed when running on Windows where there is no output per default):

```
CLCGenomicsServer_3_0.exe -q -console
```

You can also in silent mode define a different installation directory: `-dir`.

```
CLCGenomicsServer_3_0.exe -q -console -dir "c:\bioinformatics\clc"
```

Note! Both the `-console` and the `-dir` options only work when the installer is run in silent mode.

The `-q` and the `-console` options work for the Uninstall program as well.

2.5 Upgrading an existing installation

Upgrading an existing installation is very simple. For a single Genomics Server, the steps we recommend are:

- Make sure that nobody is using the server (see section 4.4). A standard procedure would be to give users advance notice that the system will be unavailable for maintenance.
- Install the server in the same installation directory as the one already installed. All settings will be maintained.

If you have a CLC job node setup, you will also need to upgrade the *CLC Science Server* software on each job node. Upgrading the software itself on each node is all you need to do. Configurations and plug-ins for job nodes are pushed to them by the master node.

2.5.1 Upgrading major versions

Once you have performed the steps mentioned above, there are a few extra details whenever the release is more than a bug-fix upgrade (e.g. a bug-fix release would be going from version 1.0 to 1.0.1).

First, make sure all client users are aware that they must upgrade their Workbench and server connection plug-in.

Second, check that all plug-ins installed on the *CLC Science Server* are up to date (see section 3.9). You can download updated versions of plug-ins from <http://www.clcbio.com/clc-plugin/>.

Third, if you are using the *CLC Server Command Line Tools*, it might have to be updated as well. This is noted in the latest improvements page of the server: http://www.clcbio.com/improvements_genomics_server.

Finally, if you are using job nodes, be aware that any new tools included in the server upgrade are automatically disabled on all job nodes. This is done in order to avoid interfering with a job node set-up, where certain job nodes are dedicated to specific types of jobs. Read more about enabling the jobs in section 6.1.3.

For major versions (e.g. going from 1.X to 2.0) a new license needs to be downloaded (see section 2.7), and the server restarted.

2.6 Allowing access through your firewall

By default, the server listens for TCP-connections on port 7777 (See section 3.4 for info about changing this).

If you are running a firewall on your server system you will have to allow incoming TCP-connections on this port before your clients can contact the server from a Workbench or web browser. Consult the documentation of your firewall for information on how to do this.

Besides the public port described above the server also uses an internal port on 7776. There is no need to allow incoming connections from client machines to this port.

2.7 Downloading a license

The CLC Science Server will look for licenses in the `licenses` folder. This means that all license files should be located in this folder. Check the platform-specific instructions below to see how to download a license file.

2.7.1 Windows license download

License files are downloaded using the `downloadlicense.command` script. To run the script, right-click on the file and choose **Run as administrator**. This will present a window as shown in figure 2.5.

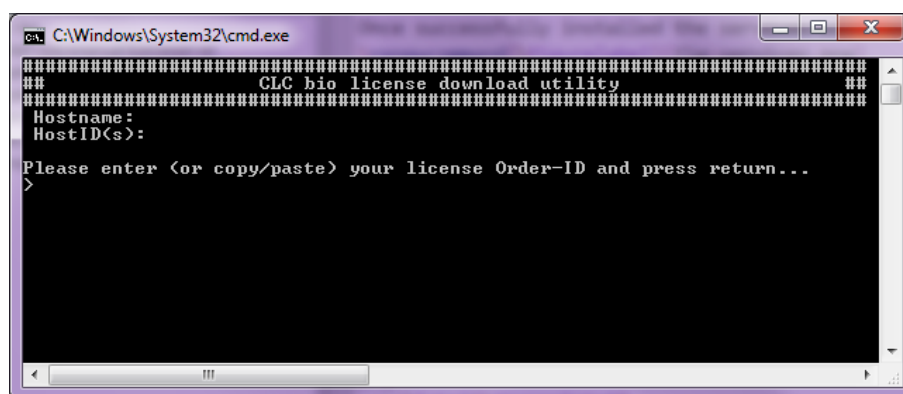


Figure 2.5: Download a license based on the Order ID.

Paste the Order ID supplied by CLC bio (right-click to **Paste**) and press Enter. Please contact support@clcbio.com if you have not received an Order ID.

Note that if you are *upgrading* an existing license file, this needs to be deleted from the `licenses` folder. When you run the `downloadlicense.command` script, it will create a new license file.

Restart the server for the new license to take effect (see how to restart the server in section 2.8.1).

2.7.2 Mac OS license download

License files are downloaded using the `downloadlicense.command` script. To run the script, double-click on the file. This will present a window as shown in figure 2.6.

Paste the Order ID supplied by CLC bio and press Enter. Please contact support@clcbio.com if you have not received an Order ID.

Note that if you are *upgrading* an existing license file, this needs to be deleted from the `licenses` folder. When you run the `downloadlicense.command` script, it will create a new license file.

Restart the server for the new license to take effect (see how to restart the server in section 2.8.2).

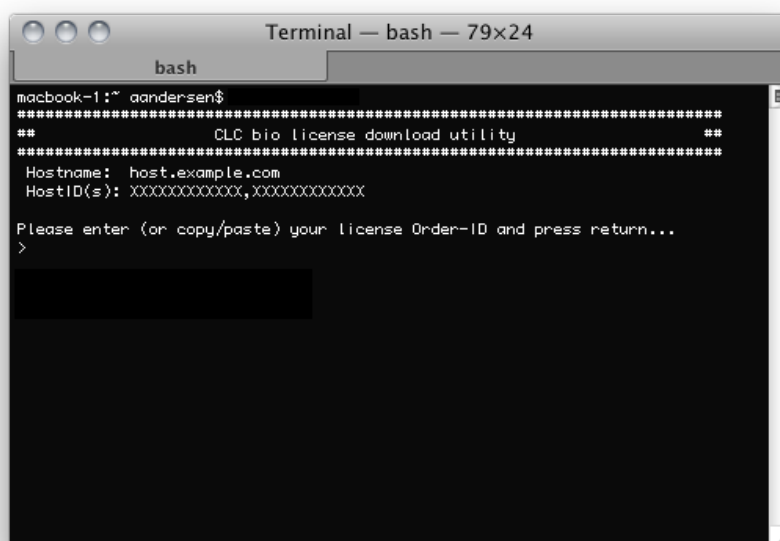


Figure 2.6: Download a license based on the Order ID.

2.7.3 Linux license download

License files are downloaded using the `downloadlicense` script. Run the script and paste the Order ID supplied by CLC bio. Please contact support@clcbio.com if you have not received an Order ID.

Note that if you are *upgrading* an existing license file, this needs to be deleted from the `licenses` folder. When you run the `downloadlicense` script, it will create a new license file.

Restart the server for the new license to take effect. Restarting the server is covered in in section [2.8.3](#).

2.8 Starting and stopping the server

2.8.1 Microsoft Windows

On Windows based systems the *CLC Science Server* can be controlled through the *Services* control panel. Choose the service called `CLCScienceServer` and click the start, stop or restart link as shown in figure [2.7](#).

2.8.2 Mac OS X

On Mac OS X the server can be started and stopped from the command line.

Open a terminal and navigate to the CLC Server installation directory. Once there the server can be controlled with the following commands.

To start the server run the command:

```
sudo ./CLCScienceServer start
```

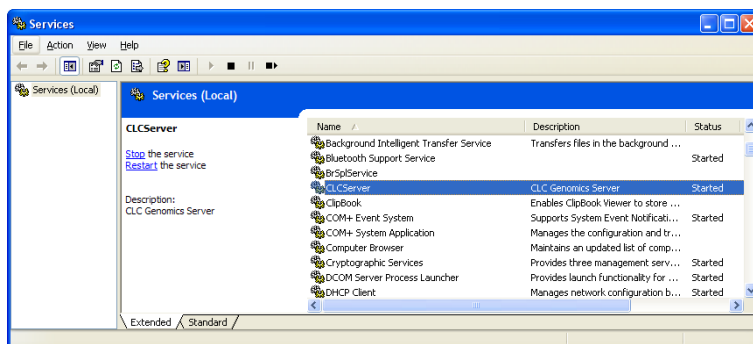


Figure 2.7: Stopping and restarting the server on Windows by clicking the blue links.

To stop the server run the command:

```
sudo ./CLCScienceServer stop
```

To view the current status of the server run the command:

```
sudo ./CLCScienceServer status
```

You will need to set this up as a service if you wish it to be run that way. Please refer to your operating system documentation if you are not sure how to do this.

2.8.3 Linux

You can start and stop the Genomics Server from the command line. You can also run it as a service.

Command Line:

Open a terminal and navigate to the CLC Server installation directory. Once there the server can be controlled with the following commands.

To start the server run the command:

```
./CLCScienceServer start
```

To stop the server run the command:

```
./CLCScienceServer stop
```

To view the current status of the server run the command:

```
./CLCScienceServer status
```

You can run these commands as any user, either directly on the command, or by prefacing the commands with `sudo`. e.g .

```
sudo -u <username> ./CLCScienceServer status
```

Please note: To run the Genomics Server as a non-root user, you should first ensure that the files under your *CLC Science Server* installation directory have the appropriate permissions. The easiest way to do this is to recursively change the ownership of all the files in that directory to the user that will run the *CLC Science Server* process.

As a service:

On installation a link is created in `/etc/init.d` to the `CLCScienceServer` script. At this point, you need to configure it as a service. For example, on Red Hat systems,

- `service clcserver-startupscript start`

We provide an example wrapper script for running the *CLC Science Server* process. You may find this convenient if you plan to run the process as a user other than the root user. The script is called `clcserver-startupscript` and can be found under the **conf** directory of the *CLC Science Server* installation area.


Please refer to your operating system documentation if you require further information about setting up services on your system.

2.9 Installing relevant plug-ins in the Workbench

In order to use the *CLC Science Server* from a CLC Workbench, you need to install the CLC Workbench Client Plug-in in the Workbench. This plug-in is needed for logging onto the server and accessing data from the server data locations.

Plug-ins are installed using the Plug-ins and Resources Manager¹, which can be accessed via the menu in the Workbench

Help | Plug-ins and Resources ()

or via the **Plug-ins** () button on the Toolbar.

From within the Plug-ins and Resources Manager, choose the Download Plug-ins tab and click on the CLC Workbench Client Plugin. Then click in the button labelled **Download and Install**.

If you are working on a system not connected to the network, then you can also install the plug-in by downloading the `cpa` file from the plugins page of our website

<http://www.clcbio.com/clc-plugin/>

Then start up the Plug-in manager within the Workbench, and click on the button at the bottom of the Plug-in manager labelled **Install from File**.

You need to restart the Workbench before the plug-in is ready for use.

Note that if you want users to be able to use **External applications** (see chapter 7) on the server, there is a separate plug-in (CLC External Applications Plug-in) that needs to be installed in the Workbench the same way as described above.

¹In order to install plug-ins on many systems, the Workbench must be run in administrator mode. On Windows Vista and Windows 7, you can do this by right-clicking the program shortcut and choosing "Run as Administrator".

Chapter 3

Configuring and administering the server

3.1 Logging into the administrative interface

Once the server is running, you can log into its administrative interface via a web browser. Most configuration occurs via this interface. Simply type the host name of the server machine you have installed the *CLC Science Server* software on, followed by the port it is listening on. Unless you change it, the port number is 7777. An example would be

```
http://clccomputer:7777/
```

The default administrative user credentials are:

```
username: root
password: default
```

Use these details the first time you log in.

3.2 Adding locations for saving data

Before you can use the server for doing analyses you will need to add one or more locations for storing your data. The locations can either be simple pointers to a folder on the file system or based on a *CLC Bioinformatics Database*.

To set up a location, open a web browser and navigate to the CLC Server web interface.

Once logged in go to the *Admin* tab and unfold the *Main configuration* section. There are two headings relating to CLC data storage: Database locations and File system locations.

3.2.1 Adding a database location

Before adding a database location, you need to set-up the database. This is described in section [2.2](#).

Under the **Database locations** heading, click the **Add New Database Location** button to add a new database location (see figure [3.2](#)).

Enter the required information about host, port and type of database. The user name and password refers to the user role on your Database Management System (DBMS), see section

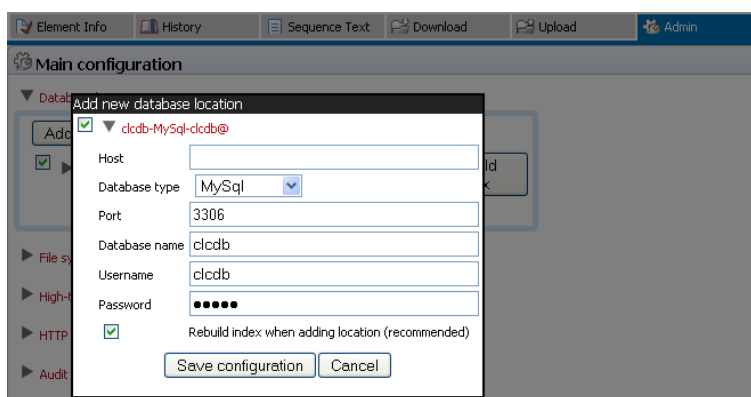


Figure 3.1: Database location settings.

2.2. Note that there are two version of Oracle in the list. One is the traditional using SID style (e.g. `jdbc:oracle:thin:@[HOST][:PORT]:SID`) and the other is using thin-style service name (e.g. `jdbc:oracle:thin:@//[HOST][:PORT]/SERVICE`).

Click the *Save Configuration* button to perform the changes. The added database location should now appear in the **Navigation Area** in the left hand side of the window.

3.2.2 Adding a file system location

Under the **File system locations** heading, click the **Add New File Location** button to add a new file system location (see figure 3.2).

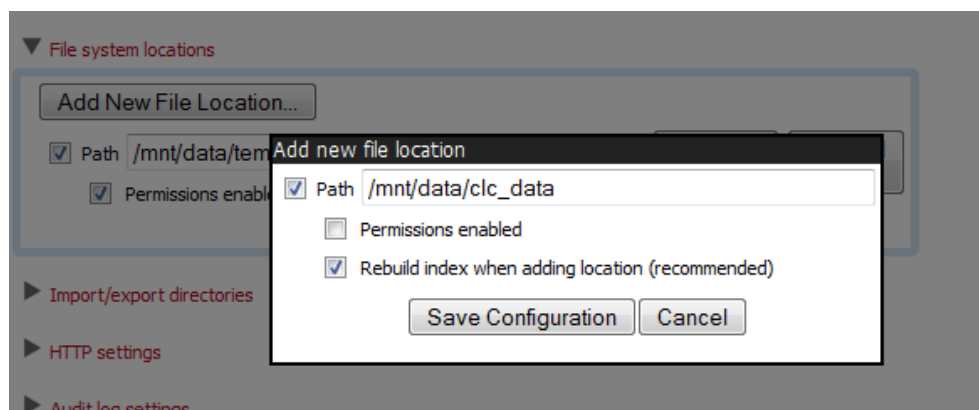


Figure 3.2: File system location settings.

In this dialog, enter the path to the folder you want to use for storing the data. The path should point to an *existing* folder on the server machine, and the user *running the server process* needs to have read and write access to the folder. This is usually a dedicated user, or it may be the system's root user if you have not created a dedicated user for this purpose.

The file location(s) configured on the server will be accessible to those working using CLC Workbenches after they log into the server via their Workbench.

Once you have pressed **Save Configuration** (learn more about rebuilding the index in section 3.2.3), this location will be added and it should now appear in the **Navigation Area** in the left hand side of the window. By default it will also appear in the Workbench on next login. You can use the checkbox next to the location to indicate whether it should be visible to your users or not.

You can choose whether access control should be switched on and off. Please see section 5.1 for more information about enabling and setting permissions on *CLC Science Server* data folders.

Note that pressing **Remove Location** will only remove the location from this list - it will not delete the folder from your system or affect any data already stored in this folder. The data will be accessible again simply by adding the folder as a new location again.

Important points about the CLC Server data in the file system locations

Any file system locations added here should be folders **dedicated for use** by the *CLC Science Server*. Such areas should be directly accessed only by the *CLC Science Server*. In other words, files should **not** be moved into these folders, or their subfolders, manually, for example using your standard operating system's command tools, drag and drop, and so on. All the data stored in this areas will be in *clc* format and will be owned by the user that runs the *CLC Science Server* process.

File locations for job node set-ups

When you have a job node set-up, all the job node computers need to have access to the same data location folder. This is because the job nodes will write files directly to the folder rather than passing through the master node (which would be a bottleneck for big jobs). Furthermore, the user running the server must be the same for all the job nodes and it needs to act as the same user when accessing the folder no matter whether it is a job node or a master node.

The data location should be added **after** the job nodes have been configured and attached to the master node. In this way, all the job nodes will inherit the configurations made on the master node.

One relatively common problem faced in this regard is *root squashing* which often needs to be disabled, because it prevents the servers from writing and accessing the files as the same user - read more about this at http://nfs.sourceforge.net/#faq_b11.

You can read further about job node setups in section 6

3.2.3 Rebuilding the index

The server maintains an index of all the elements in the data locations. The index is used when searching for data. For all locations you can choose to **Rebuild Index**. This should be done only when a new location is added or if you experience problems while searching (e.g. something is missing from the search results). This operation can take a long time depending on how much data is stored in this location.

If you move the server from one computer to another, you need to move the index as well. Alternatively, you can re-build the index on the new server (this is the default option when you add a location). If the rebuild index operation takes too long and you would prefer to move the old index, simply copy the folder called `searchindex` from the old server installation folder to the new server.

The status of the index server can be seen in the **User Statistics** pane showing information on where the index server resides and the number of locations currently being serviced.

3.3 Accessing files on, and writing to, areas of the server filesystem

There are circumstances when it is beneficial to be able to interact with (non-CLC) files directly on your server filesystem. A common circumstance would be importing high-throughput sequencing data from folders where it is stored on the same system that your *CLC Science Server* is running on. This could eliminate the need for each user to copy large sequence data files to the machine their *CLC Workbench* is running on before importing the data into a *CLC Science Server* CLC server data area. Another example is if you wish to export data from CLC format to other formats and save those files on your server machine's filesystem (as opposed to saving the files in the system your Workbench is running on).

From the administrator's point of view, this is about configuring folders that are safe for the *CLC Science Server* to read and write to on the server machine's system.

This means that users logged into the *CLC Science Server* from their Workbench will be able to access files in that area, and potentially write files to that area. Note that the *CLC Science Server* will be accessing the file system as the *user running the server process* - not as the user logged into the Workbench. This means that you should be careful when opening access to the server filesystem in this way. Thus, only folders that do not contain sensitive information should be added.

Folders to be added for this type of access are configured in the web administration interface **Admin** tab. Under **Main configuration**, open the **Import/export directories** (Figure 3.3) to list and/or add directories.

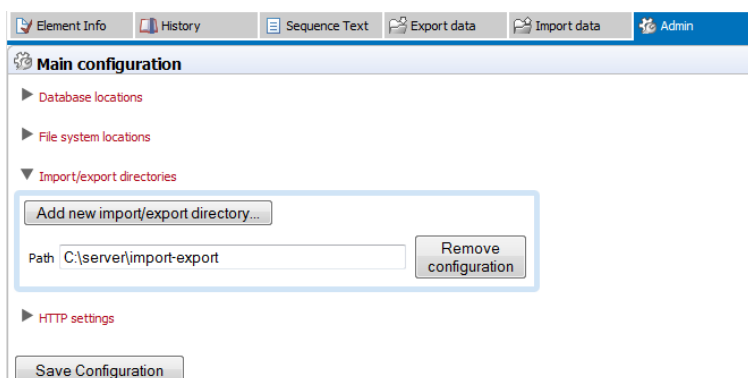


Figure 3.3: Defining source folders that should be available for browsing from the Workbench.

Press the **Add new import/export directory** button to specify a path to a folder on the server. This folder and all its subfolders will then be available for browsing in the Workbench for certain activities (e.g. importing data functions).

The import/export directories can be accessed from the Workbench via the Import function in the Workbench (Figure ??). If a user, that is logged into the *CLC Science Server* via their *CLC Workbench*, wishes to import e.g. high throughput sequencing data, an the option shown in figure 3.4 will appear.

On my local disk or a place I have access to means that the user will be able to select files from the file system of the machine their *CLC Workbench* is installed on. These files will then be transferred over the network to the server and placed as temporary files for importing. If the user chooses instead the option *On the server or a place the server has access to*, the user is presented with a file browser for the selected parts of the server file system that the administrator has configured as an Import/export location (Figure 3.5).

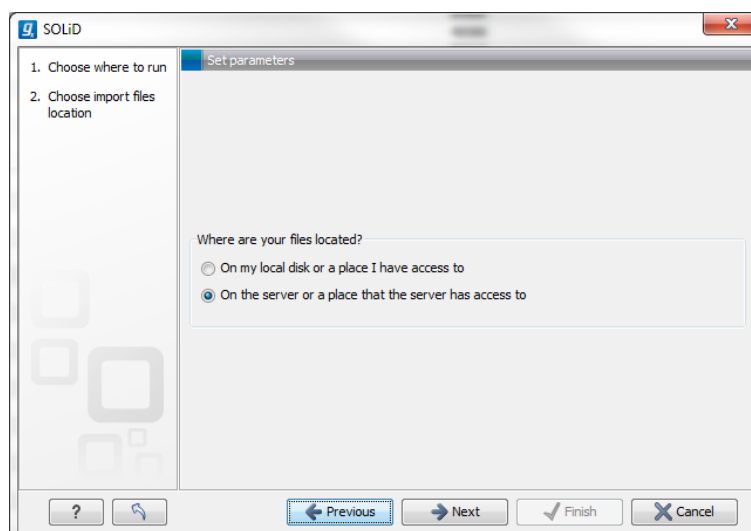


Figure 3.4: Deciding source for high-throughput sequencing data files.

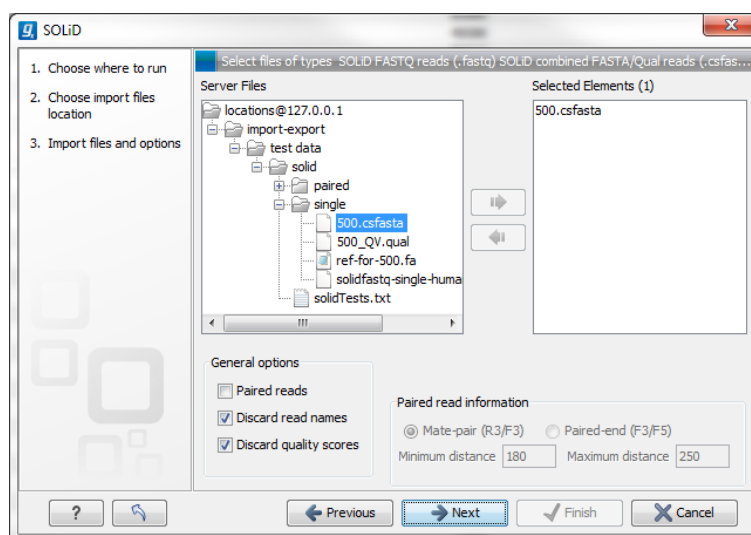


Figure 3.5: Selecting files on server file system.

Note: Import/Export locations should NOT be set to subfolders of any defined CLC file or data location. CLC file and data locations should be used for CLC data, and data should only be added or removed from these areas by CLC tools. By definition, an Import/Export folder is meant for holding non-CLC data, for example, sequencing data that will be imported, data that you export from the Genomics Server, or blast databases.

3.4 Changing the listening port

The default listening port for the CLC Server is 7777. This has been chosen to minimize the risk of collisions with existing web-servers using the more familiar ports 80 and 8080. If you would like to have the server listening on port 80 in order to simplify the URL, this can be done in the following way.

- Navigate to the CLC Server installation directory.
- Locate the file called *server.xml* in the conf directory.

- Open the file in a text editor and locate the following section

```
<Connector port="7777" protocol="HTTP/1.1"
  connectionTimeout="20000"
  redirectPort="8443" />
```

- Change the port value to desired listening port (80 in the example below)

```
<Connector port="80" protocol="HTTP/1.1"
  connectionTimeout="20000"
  redirectPort="8443" />
```

- Restart the service for the change to take effect (see how to restart the server in section 2.8).

3.5 Changing the tmp directory

The *CLC Science Server* often uses a lot of disk space for temporary files. These are files needed during an analysis, and they are deleted when no longer needed. By default, these temporary files are written to your system default temporary directory. Due to the amount of space that can be required for temporary files, it can be useful to specify an alternative, larger, disk area where temporary files created by the CLC Genomics Server can be written.

In the *server installation directory* you will find a file called `CLCScienceServer.vmoptions`. Open this file in a text editor and add a new line like this: `-Djava.io.tmpdir=/path/to/tmp` with the path to the new tmp directory. Restart the server for the change to take effect (see how to restart the server in section 2.8).

We highly recommend that the tmp area is set to a file system local to the server machine. Having tmp set to a file system on a network mounted drive can substantially affect the speed of performance.

3.5.1 Job node setups

The advice about having a tmp area being set on a local file system is true also for job nodes. Here, the tmp areas for nodes should **not** point to a shared folder. Rather, each node should have a tmp area with an identical name and path, but situated on a drive local to each node.

You will need to edit the `CLCScienceServer.vmoptions` file on each job node, as well as the master node, as described above. This setting is **not** pushed out from the master to the job nodes.

3.6 Setting the amount of memory available for the JVM

When running the *CLC Science Server*, the Java Virtual Machine (JVM) needs to know how much memory it can use. This depends on the amount of physical memory (RAM) and can thus be different from computer to computer. Therefore, the installer investigates the amount of RAM during installation and sets the amount of memory that the JVM can use.

On **Windows** and **Linux**, this value is stored in a property file called `ServerType.vmoptions` (e.g. `CLCGenomicsServer.vmoptions`) which contains a text like this:

```
-Xmx8000m
```

The number (8000) is the amount of memory the *CLC Science Server* is allowed to use. This file is located in the installation folder of the *CLC Science Server* software.

By default, the value is set to 50% of the available RAM on the system you have installed the software on.

You can manually change the number contained in the relevant line of the `vmoptions` file for your *CLC Science Server* if you wish to raise or lower the amount of RAM allocated to the Java Virtual Machine.

3.7 Limiting the number of cpus available for use

A number of the algorithms in the *CLC Science Server* will, in the case of large jobs, use all the cores available on your system to make the analysis as fast as possible. If you wish to restrict this to a predefined number of cores, this can be done with a properties file: Create a text file called `cpu.properties` and save it in the `settings` folder under the *CLC Science Server* installation directory.

The `cpu.properties` file should include one line like this:

```
maxcores = 1
```


Restart the *CLC Science Server* if you create or change this file for these settings to take effect.

Instead of 1 you write the maximum number of cores that the *CLC Science Server* is allowed to use. Please note that this is not a guarantee that the *CLC Science Server* will never use more cores than specified, but that will be for very brief and infrequent peaks and should not affect performance of other applications running on your system.

You can download a sample `cpu.properties` file at <http://clcbio.com/files/deployment/cpu.properties>.

3.8 Other configurations

3.8.1 HTTP settings

Under the **Admin**  tab, click **Configuration**, and you will be able to specify HTTP settings. Here you can set the time out for the user HTTP session and the maximum upload size (when uploading files through the web interface).

3.8.2 Audit log settings

The audit log records all the actions performed in the web interface and through the Workbenches. Note that data management operations (copying, deleting and adding files) done through the Workbench are not recorded.

3.8.3 Deployment of server information to CLC Workbenches

See the *Deployment manual* at <http://www.clcbio.com/usermanuals> for information on pre-configuring the server log-in information when Workbench users log in for the first time.

3.9 Server plug-ins

You can install plug-ins on the server under the **Admin** (⚙️) tab (see figure 3.6).

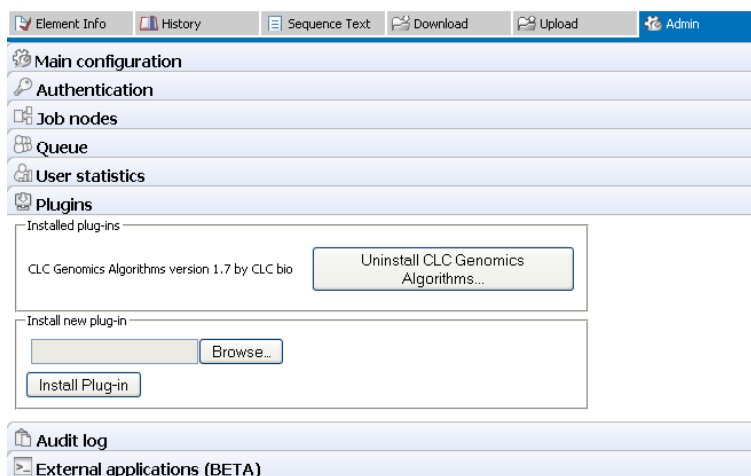


Figure 3.6: Installing and uninstalling server plugins.

Click the **Browse** button and locate the plug-in .cpa file to install a plug-in. To uninstall a plug-in, simply click the button next to the plug-in. The server does not need to be restarted after installation/uninstallation of plug-ins.

If you are running a job node setup, you **only** need to install the plug-in on the master server. However, you **need to check that the plugin is enabled for each job node** by going to the Job Distribution tab in the master nodes web administrative interface, and clicking on the link to each job node.

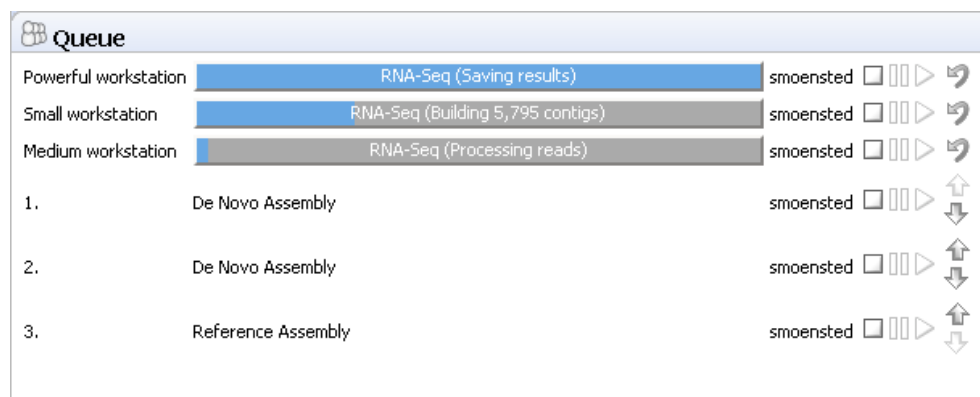
Read more about developing server plug-ins at <http://www.clcdeveloper.com/>.

3.10 Queue

Clicking the **Queue** panel will show a list of all the processes that are currently in the queue (including the one in progress). An example is shown in figure 3.7.

For each process, you are able to **Cancel** (⏏) and re-prioritize the order of the processes by clicking the up and down arrows. Some processes also allow you to pause and resume. At the top, you can see the progress of the process that is currently running.

If you are running a CLC Server with execution nodes, you can also **Stop and requeue** (🔄) a job that is currently being processed.



Queue				
Powerful workstation	RNA-Seq (Saving results)	smoensted	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
Small workstation	RNA-Seq (Building 5,795 contigs)	smoensted	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
Medium workstation	RNA-Seq (Processing reads)	smoensted	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
1.	De Novo Assembly	smoensted	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
2.	De Novo Assembly	smoensted	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
3.	Reference Assembly	smoensted	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>

Figure 3.7: *The process queue.*

Chapter 4

Managing users and groups

4.1 Logging in the first time - root password

When the server is installed, you will be able to log in via the web interface using the following credentials:

- **User name:** `root`
- **Password:** `default`

Once logged in, you should as a minimum set up user authentication (see section 4.2) and data locations (see section 3.2) before you can start using the server.

For security reasons, you should change the root password (see figure 4.1):

Admin (🔑) | Authentication (🔑) Change root password

Note that if you are going to use job nodes, it makes sense to set these up before changing the authentication mechanism and root password (see section 6).

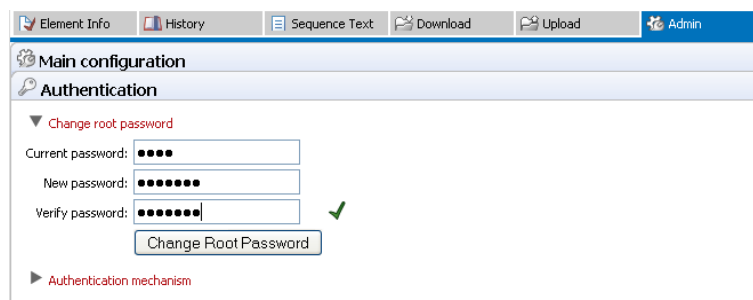


Figure 4.1: We recommend changing the root password. The verification of the root password is shown with the green checkmark.

4.2 User authentication using the web interface

When the server is installed, you can log in using the default root password (username=root, password=default). Note that if you are going to use job nodes, it makes sense to set this up first before changing the authentication mechanism and root password (see section 6).

Once logged in, you can specify how the general user authentication should be done:

Admin (🔑) | Authentication (🔑) Authentication mechanism

This will reveal the three different modes of authentication as shown in figure 4.2.

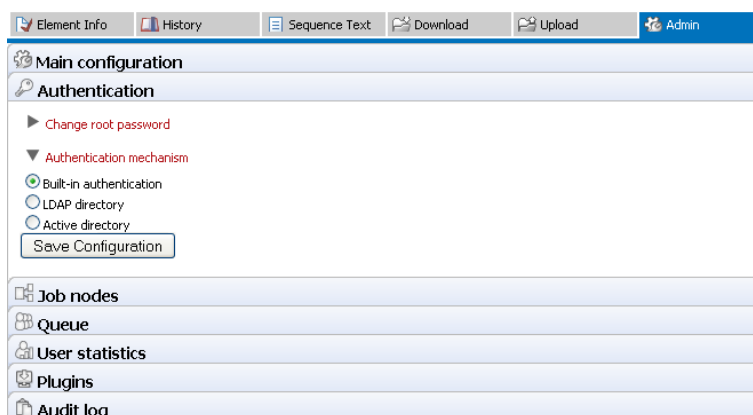


Figure 4.2: Three modes of user authentication.

The options are:

- **Built-in authentication.** This option will enable you to set up user authentication using the server's built-in user management system. This means that you create users, set passwords, assign users to groups and manage groups using the web interface (see section 4.2.1) or using the Workbench (see section 4.3.1). All the user information is stored on the server and is not accessible from other systems.
- **LDAP directory.** This option will allow you to use an existing LDAP directory. This means that all information needed during authentication and group memberships is retrieved from the LDAP directory. If needed, the LDAP integration can use Kerberos / GSSAPI.
- **Active directory.** This option will allow you to use an existing Active directory which is Microsoft's LDAP counterpart. This means that all information needed during authentication and group memberships is retrieved from the Active directory.

For the two last options, a settings panel will be revealed when the option is chosen, allowing you to specify the details of the integration.

Note that membership of an administrative group is used to control which users can access the admin part of the web interface. These users will also be able to set permissions on folders (see section 5). For the built-in authentication method, this means adding particular users to the built-in **admin** group. For Active Directory or LDAP, this means designating a group in the box labeled **Admin group name** and adding any users who should be administrators of the CLC Server to this group.

4.2.1 Managing users using the web interface

To create or remove users or change their password:

Admin (🔑) | Users and groups (👤) Manage user accounts

This will display the panel shown in figure 4.3.

The screenshot shows the 'Users and groups' web interface. The 'Manage user accounts' section is active, indicated by a red arrow. It features a large empty list box on the left. On the right, there are three panels: 'Add user account' with fields for Username, Password, and Verify password, and an 'Add User' button; 'Change password for selected user' with fields for Password and Verify password, and a 'Set Password' button; and 'Remove selected user' with a 'Remove User' button.

Figure 4.3: *Managing users.*

4.2.2 Managing groups using the web interface

To create or remove groups or change group membership for users:

Admin (🔧) | Users and groups (👤) Manage groups

This will display the panel shown in figure 4.4.

The screenshot shows the 'Users and groups' web interface. The 'Manage groups' section is active, indicated by a red arrow. The 'Manage user accounts' section is also visible but not active. The 'Manage groups' section features a list box on the left containing the user 'admin'. On the right, there are three panels: 'Create new group' with a 'Group name' field and a 'Create group' button; 'Remove selected group' with a 'Remove Group' button; and 'Manage membership' with two list boxes labeled 'Users' and 'Group members', and two buttons labeled '-'> and '<-'.

Figure 4.4: *Managing users.*

The same user can be a member of several groups.

Note that membership of the admin group is used for allowing users access to the admin part of the web interface. Users who should have access to the administrative part of the server should

be part of the "admin" group which is the only special group (this group is already created for you).

Note that you will always be able to log in as root with administrative access.

The functionality of this plug-in depends on the user authentication and management system: if the built-in system is used, all the functionality described below is relevant; if an external system is used for managing users and groups, the menus below will be disabled.

4.3 User authentication using the Workbench

Users and groups can also be managed through the Workbench (note that you need to set up the authentication mechanism as described in section 4.2):

File | Manage Users and Groups

This will display the dialog shown in figure 4.5.

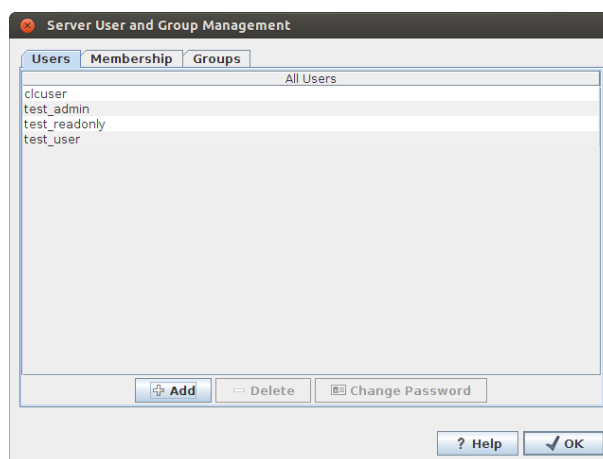


Figure 4.5: *Managing users.*

4.3.1 Managing users through the Workbench

Click the **Add** (+) button to create a new user. Enter the name of the user and enter a password. You will be asked to re-type the password. If you wish to change the password at a later time, select the user in the list and click **Change password** (key icon).

To delete a user, select the user in the list and click **Delete** (-).

4.3.2 Managing groups through the Workbench

Access rights are granted to groups, not users, so a user has to be a member of one or more groups to get access to the data location. Here you can see how to add and remove groups, and next you will see how to add users to a group.

Adding and removing groups is done in the **Groups** tab (see figure 4.6).

To create a new group, click the **Add** (+) button and enter the name of the group. To delete a group, select the group in the list and click the **Delete** (-) button.

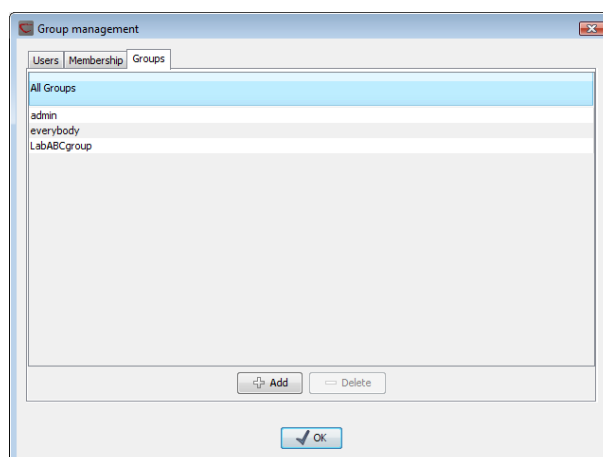


Figure 4.6: Managing groups.

4.3.3 Adding users to a group

When a new group is created, it is empty. To assign users to a group, click the **Membership** tab. In the **Selected group** box, you can choose among all the groups that have been created. When you select a group, you will see its members in the list below (see figure 4.7). To the left you see a list of all users.

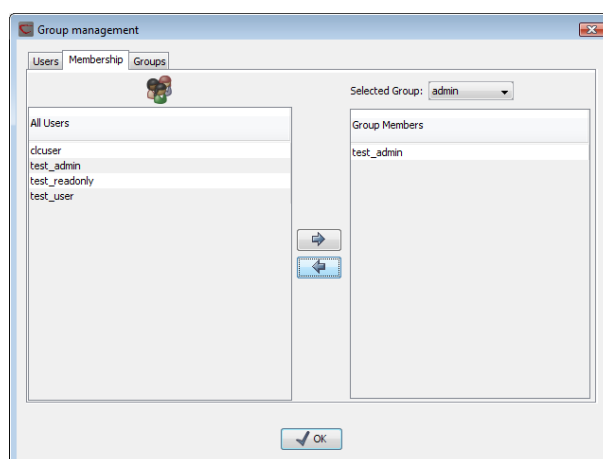


Figure 4.7: Listing members of a group.

To add or remove users from a group, click the **Add** (➡) or **Remove** (⬅) buttons. To create new users, see section 4.3.1.

The same user can be a member of several groups.

4.4 User statistics

Clicking the **User statistics** panel will show a summary of the current usage of the server. An example is shown in figure 4.8.

You can see the number of users currently logged in, and you can see the number of sessions for each user. The two green dots indicate that this user is logged in twice (e.g. through the Workbench and through the web interface). The other two users have been logged in previously.



Figure 4.8: The user statistics (user names have been blurred).

You can also log users off by expanding the user sessions on the + sign and the click **Invalidate Session....** This will open a confirmation dialog where you can also write a message to the user that will be displayed either in the Workbench or the browser.

Chapter 5

Access privileges and permissions

The *CLC Science Server* allows server administrators to control access to the server on several levels:


- Access to the data in the server's **data locations**. This is typically to allow groups of users to store data that cannot be accessed by other users, or to establish reference data sources that are "read-only" for most users.
- Access to **running jobs** on the server. Particular groups of users can be restricted to running only particular types of jobs on the server.
- Access to the **import/export directories**. The server administrator can give users access to browsing designated directories on the server file system. Thus it can be desirable to restrict this access to certain groups of users for security reasons.


The following sections describe each of these levels.

5.1 Controlling access to data

The *CLC Science Server* uses folders as the basic unit for controlling access to data, and access is granted (or denied) to groups of users.

On any folder within a location, you can grant two kinds of access to a group:

Read access This will make it possible for the users of the group to see the elements in this folder, to open them and to copy them. Access can be through any route, for example, browsing in the **Navigation Area**, searching or clicking "originates from" in the **History**  of e.g. an alignment.

Write access Changes to an element can be saved **Save** , and new elements and subfolders can be created.

For a user to be able to access a folder, there has to be at least read access to all the top folders. In the example shown in figure 5.1, to access the *Sequences* folder, the user must have at least read access to both the *Example Data* and *Protein* folders.

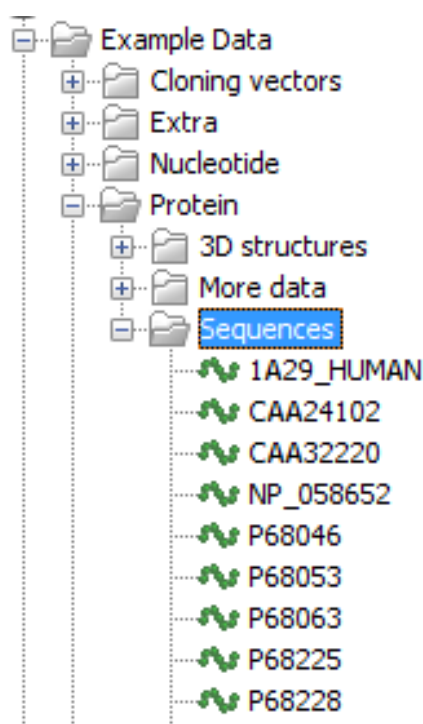


Figure 5.1: A folder hierarchy on the server.

However, you can grant read access to the *Example Data* and *Protein* folders and only grant write access to the *Sequences* folder.

Permissions on file system locations must be **explicitly enabled** if they are desired (see section 3.2.2). Please see 5.1.2 for further details about the system behaviour if permission are not enabled and configured.

If permissions are enabled on a file system location, then by default, no groups have read or write access to any area under this location until permissions are configured. Only the *CLC Science Server* root user will have access to the data held in the server at this point. In other words, you must set permissions on folders in this file system location before any users will be able to read from or write to it.

5.1.1 Setting permissions on a folder

This step is done from within a CLC Workbench. Start up a copy of a Workbench that has the CLC Workbench Client plugin installed. From within the Workbench, go to the File menu and choose the item **CLC Server Login**. Log into the CLC Server as an administrative user.

You can then set permissions on folders in your database, if you have one, or on folders within file system locations that have had permissions enabled.

right-click the folder (📁) | Permissions

This will open the dialog shown in figure 5.2.

Set the relevant permissions for each of the groups and click **OK**.

If you wish to apply the permissions recursively, that is to all subfolders, check **Apply to all subfolders** in the dialog shown in figure 5.2. **Note** that this operation is only relevant if you wish

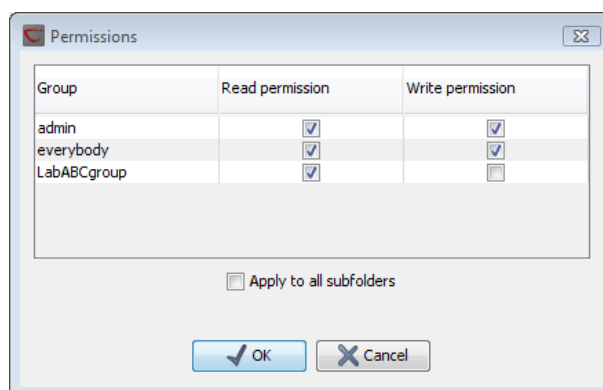


Figure 5.2: Setting permissions on a folder.

to clean-up the permission structure of the subfolders. **It should be applied with caution**, since it can potentially destroy valuable permission settings in the subfolder structure.

Permissions on the recycle bin

The recycle bin is conceptually a folder like any else. It is special in the sense that all users must have write access (otherwise they will not be able to delete anything). Because there is one recycle bin for the data from all users, you should be careful granting everybody read access to the recycle bin, since they will then be able to see the data deleted by other users. We recommend only granting read access to administrators.

Only administrators are allowed to empty the recycle bin.

5.1.2 Technical notes about permissions and security

All data stored in *CLC Science Server* file system locations are owned by the user that runs the *CLC Science Server* process. Changing the ownership of the files using standard system tools is not recommended and will usually lead to serious problems with data indexing and hamper your work on the *CLC Science Server*.

One implication of the above ownership setup is that by default, (i.e. without permissions enabled), all users logging into the *CLC Science Server* are able to access all data within that file system location, and write data to that file system locations. All files created within such a file system location are then also accessible to all users of the *CLC Science Server*.

Group permissions on file system locations is an additional layer within the *CLC Science Server*, and is not part of your operating system's permission system. This means that enabling permissions, and setting access restrictions on CLC file system locations only affects users accessing data through CLC tools (e.g. using a Workbench, the CLC Command Line Tools, the *CLC Science Server* web interface or the Server API). If users have direct access to the data, using for example general system tools, the permissions set on the data in *CLC Science Server* has no effect.

5.2 Global permissions

In the server web interface, in the **Admin tab** you can set global permissions for the *CLC Science Server* as shown in figure 5.3.

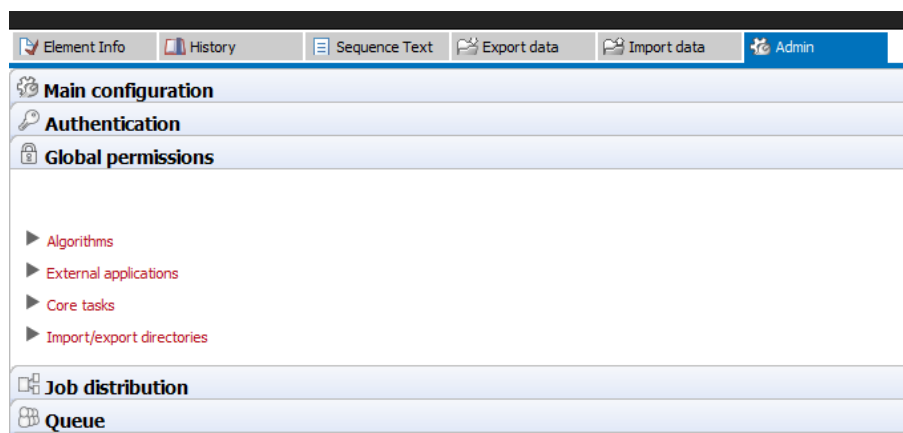


Figure 5.3: Global permissions.

Permissions can be set for:

- **Algorithms** All the algorithms running on the server are listed here.
- **External** applications. All the external applications configurations are listed here.
- **Core tasks** These are general import, export and maintenance tasks.
- **Import/export directories** Permissions can be set for each of the directories defined.

You can specify which groups should have access by clicking the **Edit Permissions** button. A dialog will appear like that in figure 5.4. If you choose **Only authorized users from selected groups**, you will be offered a list of groups that you can select (or de-select) to give (or take away) access to that functionality.

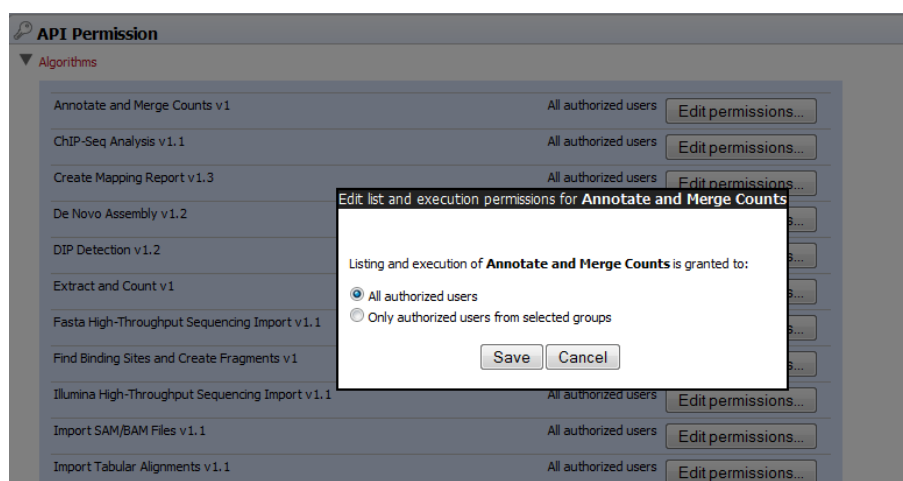


Figure 5.4: Setting permissions for an alorithm.

The default configuration is that all users have access to everything.

Chapter 6

Job Distribution

The *CLC Science Server* has the concept of *distributing jobs to nodes*. This means that you can have a master server with the primary purpose of handling user access, serving data to users and starting jobs, and you have a number of nodes that will execute these jobs.

Two main models of this setup are available: a master server that submits tasks to dedicated execution nodes, and a master server that submits tasks to a local grid system. Figure 6.1 shows a schematic overview of these possibilities, and they are described in text below.

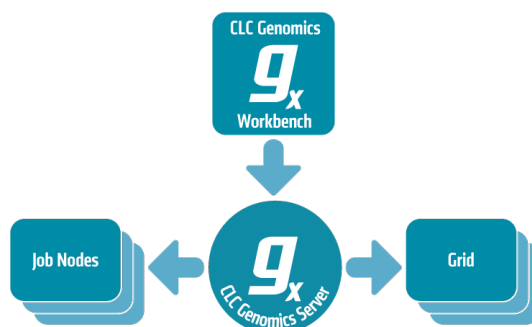


Figure 6.1: An overview of the job distribution possibilities.

- **Model I: Master server with execution nodes** - here, a master server submits CLC jobs directly to machines running the *CLC Science Server* for execution. In this setup, a group of machines (from two upwards) have the *CLC Science Server* software installed on them. The system administrator assigns one of them as the *master node*. This node controls the distribution of jobs. The other nodes are *execution nodes*, which carry out the computational tasks they are assigned. The execution nodes wait for the master node to assign them a job, and once the job is executed, they are available for the next job. With this set-up, the CLC master server controls the queue and the distribution of compute resources. This has the advantage of being simple to set up and maintain, with no other software required. However, it is not well suited to situations where the compute resources are shared with other systems because there is no mechanism for managing the load on the computer. This setup works best when the execute nodes are machines dedicated to running a *CLC Science Server*. Further details about this setup can be found in section 6.1
- **Model II: Master server submitting to grid nodes** - here the master server submits jobs to a third party job scheduler. That scheduler controls the resources on a local computer

cluster (grid) where the job will be executed. This means that it is the responsibility of the native grid job scheduling system to start the job. When the job is started on one of the grid nodes, a **CLC Grid Worker**, which is a stand-alone executable including all the algorithms on the server, is started with a set of parameters specified by the user. Further details about this setup can be found in section 6.2.

6.1 Model I: Master server with execution nodes

The general steps to setting up this model are to:

1. Install the *CLC Science Server* software on all the machines involved. (See section 2.3.)
2. Start up the *CLC Science Server* software on all the machines involved. (See section 2.8.)
3. Install the license on the machine that will act as the master node. (See section 2.7.)
4. Log in to the web administrative interface for the *CLC Science Server* of the machine that will act as the master node. (See section 3.1.)
5. Configure the Master node and the job nodes via the administrative interface on the Master node.

The only work you have to do directly on the machines that will run as job nodes is to install the *CLC Science Server* and start the software up on each of them. Almost all other configurations are done via the web administrative interface for the *CLC Science Server* of the master node. This includes the installation of plug-ins. See section 6.1.4.

6.1.1 Master and job nodes explained

A machine running the *CLC Science Server* software can be considered to be of one of three types:

1. **Single server** - a server, to which users submit jobs, and which runs those jobs.
2. **Master node** - a machine that accepts jobs from users and then passes them to other systems - either to execution nodes (see below) or a local grid system.
3. **Execution node** - here used to describe a machine running the *CLC Science Server* that accepts jobs directly from a Master node.

Figure 6.2 shows the configuration options for the types of machines running the *CLC Science Server*.

6.1.2 User credentials on a master-job node setup

If you have a brand new installation, and you plan to use the default administrative login credentials (see section 3.1), you do not need to change anything.

If you wish to set other authentication details, then log into the web administration interface on each machine and set up *identical* authentication details on each one.

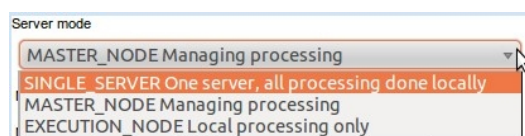


Figure 6.2: The configuration options for the types of machines running the **CLC Science Server**. The choices of relevance under normal circumstances are *Single_server* and *Master_node*. An administrator will not usually need to manually choose the *Execution Host* option. This option is there primarily to allow for troubleshooting.

You can now log out of the *CLC Science Server* web administrative interface on all the machines except the one that will be designated the master node. All further configuration will take place on the machine to be the master node.

6.1.3 Configuring your setup

If you have not already, please download and install your license to the master node. (See section 2.7.) Do **not** install license files on the job nodes. The licensing information, including how many job nodes you are can run, are all included in the license on the master node.

To configure your master/execution node setup, navigate through these tabs in the web administrative interface on your master node:

Admin (⚙️) | **Job distribution** (👤)

First, set the server mode to `MASTER_NODE` and provide the master node address, port and a human-readable name as shown in figure 6.3.

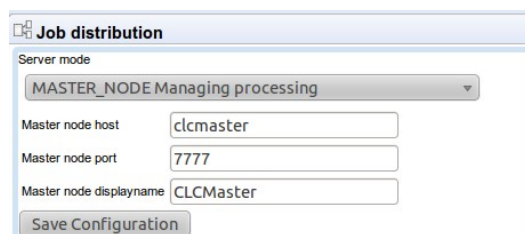


Figure 6.3: Setting up a master server.

Next, click **Attach Node** to specify a job node. Fill in the appropriate information about the node (see figure 6.4).

Besides information about the node hostname, port and displayname, you can also configure what kind of jobs that node is able to execute.

Repeat this process for each job node you wish to attach and click **Save Configuration** when you are done.

Once set up, the job nodes will automatically inherit all configurations made on the master node. At any time, you can log in to a job node itself via its Server administrative interface.

Note that you will get a warning dialog if there are types of jobs that are not enabled on any of the nodes.

Note that when a node has finished a job, it will take the first job in the queue that is of a type

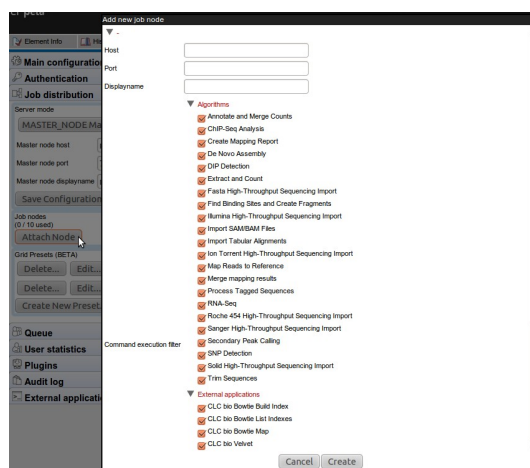


Figure 6.4: Setting up a master server.

the node is configured to process. This then means that, depending on how you have configured your system, the job that is number one in the queue will not necessarily be processed first.

After your job nodes are all configured, file system locations should be added (see the section on Adding a file system location 3.2.2). We recommend this is done at this point as all job nodes will then inherit the configurations made on the master node.

In order to test that access works for both job nodes and the master node, you can perform the following check:

1. Create a new folder in the Workbench in the new data location (on disk, this will be done by the master node). Import some data into this folder.
2. In the Workbench, run one of the analyses in the server toolbox and choose to save the results in the new folder (this will be done by the job node)
3. Once finished, try to rename one of the result files in the Workbench (on disk, this will be done by the master node). If this works, it means that both master and job nodes are able to write to the same files.

One relatively common problem that can arise here is *root squashing*. This often needs to be disabled, because it prevents the servers from writing and accessing the files as the same user - read more about this at http://nfs.sourceforge.net/#faq_b11.

6.1.4 Installing Server plug-ins

Server plugin installation is described in section 2.3.

You **only** need to install the plug-in on the master server. Once installed, you should **check that the plugin is enabled** for **each job node** you wish to make available for users to run that particular task on. To do this:

- Go to the Job Distribution tab in the master nodes web administrative interface
- Click on the link to each job node
- Click in the box by the relevant task, marking it with a check mark if you wish to enable it.

6.2 Model II: Master server submitting to grid nodes

The CLC Grid Integration Tool allows jobs to be offloaded from a master server onto grid nodes using the local grid submission/queuing software to handle job scheduling and submission.

At the moment, a given CLC algorithm will run on a single machine. That is, a single job will run on one node. A single job is *not* distributed across nodes. Thus, each grid node employed for a given task must have enough memory and space to carry out that entire task.

6.2.1 Requirements for CLC Grid Integration

- A functional grid submission system must already be in place. Please also see the section on supported job submission systems below.
- The DRMAA library for the grid submission system to be used. Note that OGE comes with this library, while you will need to get the code for the PBS DRMAA library and compile this yourself. (Additional notes below.)
- The *CLC Science Server* must be installed on a Linux based system configured as a *submit host* in the grid environment.
- The user running the *CLC Science Server* process is seen as the submitter of the grid job, and thus this user must exist on all the grid nodes.
- CLC Genomic Server file locations holding data that will be used must be mounted with the same path on the grid nodes as on the master Genomics Server and accessible to the user that runs the CLC Genomics Server process.
- If a CLC Bioinformatics Database is in use, all the grid nodes must be able to access that database using the user that runs the CLC Genomics Server process.
- A *CLC License Server* with one or more available *CLC Genomics Grid Worker* licenses must be reachable from the *execution hosts* in the grid setup.
- A SUN/Oracle Java Runtime environment 1.6 must be installed on all execution hosts that will be running CLC Grid Worker jobs.

Supported job submission systems

The scheduling systems CLC officially supports are OGE and PBS Pro, as these are the ones we have tested.

On a more general level:

- The grid integration in the CLC Genomics Server is done using DRMAA. Integrating with any submission system that provides a working DRMAA library should be possible.
- The scheduling system must also provide some means of limiting the number of CLC jobs launched for execution so that when this number exceeds the number of CLC Gridworker licenses, excess tasks are held in the queue until licenses are released. In OGE for example, the number of simultaneous CLC jobs sent for execution on the cluster can be controlled in this way by configuring a "Consumable Resource".

An example of a system that works for submitting CLC jobs, but which cannot be officially supported due to the second of the above points is PBS Torque. As far as we know, there is no way to limit the number of CLC jobs sent simultaneously to the cluster to match the number of CLC Gridworker licenses. So, with PBS Torque, if you had three Gridworker licenses, up to three jobs could be run simultaneously. However, if three jobs are already running and you launch a fourth job, then this fourth job will fail because there would be no license available for it.

This limitation can be overcome, allowing you to work with systems such as PBS Torque, if you control the job submission in some other way so the license number is not exceeded. One possible setup for this is if you have a one-node-runs-one-job setup. You could then set up a queue where jobs are only sent to a certain number of nodes, where that number matches the number of CLC Gridworker licenses you have.

DRMAA for PBS Pro

: Code for this can be downloaded from <http://sourceforge.net/projects/pbspro-drmaa/>. When configuring the library, one has to pass an `"-with-pbs="` argument, which points to the prefix of the PBS installation root. The configure script expects to be able to find `lib/libpbs.a` and `include/pbs_ifl.h` in the given root (amongst other files). SSL is needed. The configure script expects that linking with `"ssl"` will work, thus `libssl.so` must be present in one of the system's library paths. On Red Hat and SUSE you will have to install `openssl-devel` packages to get that symlink (or create it yourself). The install procedure will install `libdrmaa.so` to the provided prefix (configure argument), which is the file the *CLC Science Server* needs to know about. The PBS DRMAA library can be configured to work in various modes as described in the README file of the `pbs-drmaa` source code. We have experienced the best performance, when the CLC Server has access to the PBS log files and `pbs-drmaa` is configured with `wait_thread 1`.

6.2.2 Technical overview

Figure 6.5 shows an overview of the communication involved in running a job on the grid, using OGE as the example submission system.

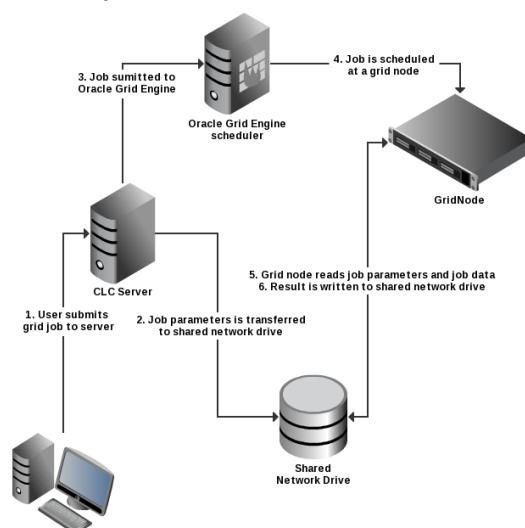


Figure 6.5: An overview of grid integration, using OGE as the example submission system.

The steps of this figure are in detail:

1. From the Workbench the user invokes an algorithm to be run on the grid. This information is sent to the master server running the *CLC Science Server*.
2. The master server writes a file with job parameters to shared work directory of the grid execution nodes. The job parameters contain identifiers mapping to the job data placed in the CLC server data location. The job parameters file is automatically deleted when it is no longer used by the grid node.
3. Now the server invokes *qsub* through the specified DRMAA native library. Then *qsub* transfers the job request to the grid scheduler. Since the user that runs the CLC Server process has invoked *qsub*, the grid node will run the job as this CLC-Server user.
4. The job scheduler will choose a grid node based on the parameters given to *qsub* and the user that invoked *qsub*.
5. The chosen grid node will retrieve CLC Grid Worker executable and the job parameters from the shared file system and start performing the given task.
6. After completion of the job, the grid node will write the results to the server's data location. After this step the result can be accessed by the Workbench user through the master server.

6.2.3 Setting up the grid integration

CLC jobs are submitted to a local grid via a special, stand-alone executable called **clcgridworker**. In the documentation, this executable is also referred to as the CLC Grid Worker.

The following steps are taken to setup grid integration for CLC bio jobs. These steps are described in more detail in the sections that follow. It is assumed that your CLC Genomics Server software is already installed on the machine that is to act as the master.

1. Set up the licensing of the grid workers, as described in section 6.2.4
2. Configure the CLC grid licenses as a consumable resource in the local grid system, as described in section 6.2.5
3. Configure and save grid presets, as described in section 6.2.6
4. If not already done, install the CLC Workbench Client Plugin in the Workbenches, as described in section 2.9 to be used to submit jobs to your grid.
5. Optionally, create and edit a `clcgridworker.vmoptions` file in each deployed CLC Grid Worker area, as described in section 6.2.8. This is usually desirable and would be done if you wished to customize settings such as maximum memory to dedicate to the java process.
6. Test your setup by submitting some small tasks to the grid via a *CLC Science Server* client such as the CLC Genomics Workbench or the Command Line Tools. Ideally, these would be tasks already known to run smoothly directly on your *CLC Science Server*.

6.2.4 Licensing of grid workers

There are two main steps involved in setting up the licenses for your CLC Grid Workers.

Step 1: Installing network licenses and making them available for use

Generally, a pool of CLC grid licenses are purchased and are served by the *CLC License Server* software. For information on how to install the *CLC License Server* and download and install your CLC Grid Worker licenses please follow the instructions in the *CLC License Server* user manual, which can be found at

<http://www.clcsupport.com/clclicensesserver/current/>.

A pdf version is available at

http://www.clcbio.com/files/usermanuals/CLC_License_Server_User_Manual.pdf.

Step 2: Configuring the location of your CLC License Server for your CLC Grid Workers

One license is used for each CLC Grid Worker script launched. When the CLC Grid Worker starts running on a node, it will attempt to get a license from the license server. Once the job is complete, the license will be returned. Thus, your CLC Grid Worker needs to know where it can contact your CLC License Server to request a license.

To configure this, use a text editor and open the file: `gridres/settings/license.properties` under the installation of your CLC Genomics Server.

The file will look something like this:

```
#License Settings

serverip=host.example.com
serverport=6200
disableborrow=false

autodiscover=false
useserver=true
```

You can leave `autodiscover=true` to use UDP-based auto discovery of the license server. However, for grid usage it is recommended that you set `autodiscover=false` and use the `serverip` property to specify the host name or IP-address of your CLC License Server.

After you have configured your grid presets, see section 6.2.6, and have saved them, those presets are deployed to the location you specify in the **Path to CLC GridWorker** field of the preset. Along with the `clcgridworker` script, the license settings file is also deployed.

If you need to change your license settings, we recommend that you edit the `license.properties` file under `gridres/settings/license.properties` of your *CLC Science Server* installation, and then re-save each of your grid presets. This will re-deploy the CLC Grid Workers, including the changed `license.properties` file.

6.2.5 Configuring licenses as a consumable resource

Since there is a limitation on the number of licenses available, it is important that the local grid system is configured so that the number of CLC Grid Worker scripts launched is never higher than the maximum number of licenses installed. If the number of CLC Grid Worker scripts launched exceeds the number of licenses available, jobs unable to find a license will fail when they are executed.

Some grid systems support the concept of a *consumable resource*. Using this, you can set the number of CLC grid licenses available. This will restrict the number of CLC jobs launched to run on the grid at any one time to this number. Any job that is submitted while all licenses are already in use should sit in the queue until a license becomes available. **We highly recommend that CLC grid licenses are configured as a consumable resource on the local grid submission system.**

6.2.6 Configure grid presets

The details of the communication between the master server and the grid when submitting a job is configured using **grid presets**. The users selects a preset when starting the job as explained in section 6.2.10.

To configure the presets, log into the web-interface of the *CLC Science Server* on your master machine and navigate through these tabs in the web administrative interface:

Admin (🔧) | Job distribution (🖨️)

Choose the **Grid Presets** section and click the **Create New Preset** button.

Figure 6.6: Configuring presets.

For each preset, the following information can be set:

Preset name The name of the preset as it will be presented in the Workbench when starting a job (see section 6.2.10) and as you will be able to refer to it when using the Command Line Tools. Alphanumeric characters can be used, and hyphens are fine within, but not at the start of, preset names.

Native library path The full path to the grid-specific DRMAA library.

Shared work directory The path to a directory that can be accessed by both the CLC Server and the Grid Workers. Temporary directories are created within this area during each job run.

These temporary directories hold files used for communication between the CLC Server and Grid Worker.

Path to CLC Grid Worker This field should contain the path to a directory on a shared file system that is readable from all execution hosts.

The CLC Grid Worker, along with associated settings files, is extracted from the installation area of the *CLC Science Server* software, and is then deployed to this location when you save your grid preset, or whenever you update plugins on your system.

If this directory does not exist, it will be created.

In versions of the Genomics Server earlier than 5.0, this path needed to point at the `clcgridworker` script itself. To support backwards compatibility with existing setups, we ask that you do **not** use the name "clcgridworker" for a directory you wish your CLC Grid Worker to be deployed to.

Job category The name of the job category - a parameter passed to the underlying grid system.

Native specification List of native parameters to pass to the grid e.g. associations to specific grid queues or memory requirements (see below). Clicking on the `f(x)` next to Native Specification pops up a box allowing the insertion of particular variables into the Native Specification box. This is described further below [6.2.6](#)

Below are examples of OGE-specific arguments one might provide in the native specification field of a Grid Preset. Please see your grid scheduling documentation to determine what options are available for your scheduling system.

Example 1: To redirect standard output and error output, you might put the following in the Native specification field:

```
-o <path/to/standard_out> -e <path/to/error_out>
```

This corresponds to the following `qsub` command being generated:

```
qsub my_script -o <path/to/standard_out> -e <path/to/error_out>
```

Example 2: Use a specific OGE queue for all jobs.

```
-hard -l qname=<name_of_queue>
```

This corresponds to the following `qsub` command:

```
qsub my_script -q queue_name
```

f(x) - adding variables evaluated at run-time

Grid Presets are essentially static in nature, with most options being defined directly in the preset itself. In some cases though, it may be of interest to have variables that are evaluated at runtime. Currently, three such variables can be added to the Native Specification line:

USER_NAME The name of the user who is logged into the server and is submitting the analysis request. All grid jobs are submitted by the user that runs the CLC Server process, so this variable might be added to, for example, log usage statistics for actual users of the system, or to send an email to the an email account of a form that includes the contents of this variable. For example, the type of text that follows could be put into the Native specification field:

```
-M {USER_NAME}@yourmailserver.com
```

COMMAND_NAME The name of the *CLC Science Server* command to be executed on the grid by the `clcgridworker` executable.

COMMAND_ID The ID of the *CLC Science Server* command to be executed on the grid.

These variables can be added by the administrator directly into the Native Specification box, by surrounding the variable name with curly brackets. Alternatively, to ensure the proper syntax, you can click on the `f(x)` link and choose the variable to insert.

These variables can be used by the administrator in any way that fits with the native grid system, and that does not cause clashes with the way the CLC Server and Grid Workers communicate. For example, in cases where grid usage is closely monitored, it may be desirable to include the user name of the analysis job in metadata reports, so that computer resource time can be logged. Another example might be to set an option such as `-q {COMMAND_NAME}` if there were, for example, certain commands to be submitted to queues of the same name as the commands.

6.2.7 Controlling the number of cores utilized

In order to configure core usage, the native specification of the grid preset needs to be properly configured. This configuration depends on the grid system used. From version 4.01 of the CLC Genomics Server, all cores on an execution node will be used by default. Unless otherwise configured to limit the number of cores used for a job involving assembly or read mapping phases, a dedicated queue must then be setup, which only schedules a single job on any given machine at a time. Otherwise your CLC jobs may conflict with others running on the same execution host at the same time.

Configuration of OGE/SGE

1) CPU Core usage when not using parallel environment

By default the CLC Genomics Servers ignores the number of slots assigned to a grid job, and utilizes all cores of the execution host. That is, jobs will run on all cores of a execution host.

As of version 4.01 of the CLC Genomics Server, there is an environmental variable, which, when set to 1, will specify that the number of allocated slots should be interpreted as the maximum number of cores a job should be run on. To set this environmental variable, add the following to the native specification of the grid preset:

```
-v CLC_USE_OGE_SLOTS_AS_CORES=1
```

In this case, the number of utilized cores is equal to the number of slots allocated by OGE for the job.

2) Limiting CPU core usage by utilizing parallel environment

The parallel environment feature can be used to limit the number of cores used by the CLC Genomics Server, when running jobs on the grid. The syntax in the native specification for using parallel environments is:

```
-pe $PE_NAME $MIN_CORE-$MAX_CORE
```

When the parallel environments feature is used, the number of allocated slots is interpreted as the number of cores to be used. That is, the number of utilized cores is equal to the number of slots in this case.

The parallel environment, selected by its name, must be setup by the grid administrator (documentation provided by Oracle will cover this subject area), in such a way that the number of slots corresponds to the number of cores. `$MIN_CORE` and `$MAX_CORE` specify a range of cores, which the jobs submitted through this grid preset can run under. Care must be taken not to set `$MIN_CORE` too high, as the job might never be run (e.g. if there is no system with that many cores available), and the submitting user will not be warned by this fact.

An example of a native specification using parallel environments is the following:

```
-l cfl=1 -l qname=32bit -pe clc 1-3.
```

Here, the `clc` parallel environment is selected, and 1 to 3 cores are requested.

Older versions of the CLC Genomics Server

CLC Genomics Server version 4.0 and older utilize CPU cores equal to the number of allocated slots, unless a parallel environment is in use, in which case the behaviour is the same as described previously. In many situations the number of allocated slots is 1, effectively resulting in CLC jobs running on one core only.

Configuration of PBS Pro

With PBS Pro it is not possible to specify a range of cores (at least not to our knowledge). Here one specifies exactly how many cores are needed. This request can be granted (the process is scheduled) or denied (the process is not scheduled). It is thus very important to choose a realistic number. The number of cores are requested as a resource: `-l nodes=1:ppn=X`, where X is the number of cores. As this resource is also designed to work with parallel system, the number of nodes is allowed to be larger than 1. For the sake of scheduling cores, it is vital that this parameter is kept at 1. An example of a native specification is: `-q bit32 -l nodes=1:ppn=2`, which will request two cores and be queued in the `bit32` queue.

6.2.8 Other grid worker options

Additional java options can be set for grid workers by creating a file called `clcgridworker.vmoptions` in the same folder as the *deployed* `clcgridworker` script, that is, the `clcgridworker` script within the folder specified in the **Path to CLC GridWorker** field of the grid preset.

For example, if a `clcgridworker.vmoptions` was created, containing the following two lines, it would, for the CLC Grid Worker specified in a given preset, set memory limits for the CLC Science Server java process and a temporary directory available from the grid nodes, overriding the defaults that would otherwise apply:

```
-Xmx1000m
-Djava.io.tmpdir=/path/to/tmp
```

For each grid preset you created, you can create a `clcgridworker.vmoptions` file within the folder you specified in the **Path to CLC GridWorker** field. So for example, if you had two grid presets, you could set two quite different memory limits for the *CLC Science Server* java process. This might be a useful idea in the case where you wished to provide two queues, one for tasks with low overheads, such as imports and trimming jobs, and one for tasks with higher overheads, such as de novo assemblies or read mappings.

6.2.9 Testing a Grid Preset

There are two types of tests that can be run to check a Grid Preset. The first runs automatically whenever the *Save Configuration* button in the Grid Preset configuration window is pressed. This is a basic test that looks for the native library you have specified. The second type of test is optional, and is launched if the *Submit test job...* button is pressed. This submits a small test job to your grid and the information returned is checked for things that might indicate problems with the configuration. While the job is running, a window is visible highlighting the jobs progression as shown in figure 6.7.

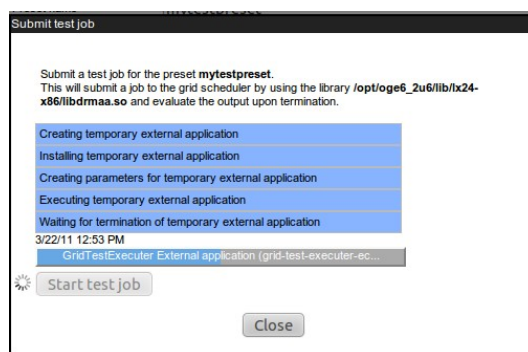


Figure 6.7: Testing a Grid Preset.

6.2.10 Client-side: starting CLC jobs on the grid

Installing the CLC Workbench Client Plug-in

To submit jobs to the grid from within a CLC Workbench, users must be able to access the *CLC Science Server*, which means that the CLC Workbench Client Plug-in must be installed in the Workbench, as described in section 2.9.

Starting grid jobs

Once the server side is configured, and the CLC Workbench Client Plug-in has been installed in the *CLC Genomics Workbench*, an extra option will appear in the first dialogue presented when setting up a task that could be executed on the *CLC Science Server*. Users will be able to choose to execute such a task on their local machine, the CLC Server machine, or using any available grid presets. To submit to the grid is as simple as choosing from among the grid presets in the drop down box. See figure 6.8.

6.2.11 Grid Integration Tips

If you are having problems with your CLC Grid Integration, please check the following points:

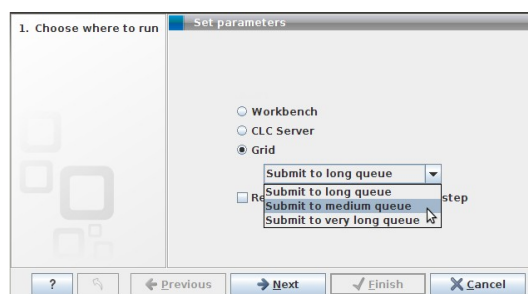


Figure 6.8: Starting the job on the grid.

- Does your system meets the requirements of the CLC Grid Integration Tool 6.2.1? For example, please check that the machine the *CLC Science Server* is running on is configured as a submit host for your grid system, and please check that you are running Sun/Oracle Java 1.6 on all execution hosts.
- The user running the *CLC Science Server* process is the same user seen as the submitter of all jobs to the grid. Does this user exist on your grid nodes? Does it have permission to submit to the necessary queues, and to write to the shared directories identified in the Grid Preset(s) and any `clcgridworker.vmoptions` files?
- Are your CLC Genomic Server file locations mounted with the same path on the grid nodes as on the master Genomics Server and accessible to the user that runs the CLC Genomics Server process?
- If you store data in a database, are all the grid nodes able to access that database, using the user account of the user running the *CLC Science Server* process?
- If you store data in a database, did you enter a machine name in the Host box of the Database Location field when setting up the Database Location using the *CLC Science Server* web administration form? In particular, a generic term such as `localhost` will not work, as the grid nodes will not be able to find the host with the database on it using that information.
- If you installed the *CLC Science Server* as root, and then later decided to run it as a non-privileged user, please ensure that you stop the server, recursively change ownership on the *CLC Science Server* installation directory and any data locations assigned to the CLC Server. Please restart the server as the new user. You may need to re-index your CLC data locations (section 3.2.3) after you restart the server.
- Is your java binary on the PATH? If not, then either add it to PATH, or edit the `clcgridworker` script in the *CLC Science Server* installation area, with the relative path from this location: `gridres/dist/clcgridworker`, and set the JAVA variable to the full path of your java binary. Then re-save each of your grid presets, so that this altered `clcgridworker` script is deployed to the location specified in the **Path to CLC GridWorker** field of your preset.
- Is the `SGE_ROOT` variable set early enough in your system that it is included in the environment of services? Alternatively, did you edit the Genomics Server startup script to set this variable? If so, the script is overwritten on upgrade - you will need to re-add this variable setting, either to the startup script, or system wide in such a way that it is available in the environment of your services.

- Is your java 64-bit, while your DRMAA library is 32-bit, or vice versa? These two things must be either both for 64-bit systems or both for 32-bit systems.

6.2.12 Understanding memory settings

Most work done by the CLC Genomics Server is done via its java process. However, particular tools involving de novo assembly or mapping phases (e.g. read mappings, RNA-seq analyses, smallRNA analyses, etc.) use C binaries for the computational phases.

Java process

For the grid worker **java process**, if there is a memory limit set in your clcgridworker.vmoptions file, this is the value that will be used. See section 6.2.8.

If there is no memory setting in your grid worker's clcgridworker.vmoptions file, then the following sources are referred to, in the order stated. As soon as a valid setting is found, that is the one that will be used:

1. Any virtual memory settings given in the grid preset, or if that is not set, then
2. Any physical memory settings given in the grid preset, or if that is not set, then
3. Half of the total memory present, with 50GB being the maximum set in this circumstance.

Please note that any nodes running a 32-bit operating system will have a maximum memory allocation for the java process of 1.2GB.

C binaries

For the computationally intensive tools that include a phase using a C binary, (e.g. de novo assembly, and jobs involving mapping phases (e.g. read mappings, RNA-seq analyses, smallRNA analyses, etc.)), the C binary phase is not restricted by the amount of memory set for the java process. For this reason, we highly recommend caution if you plan to submit more jobs of these types to nodes that are being used simultaneously for other work.

Chapter 7

External applications

Command-line applications on the server machine can easily be made available via the graphical menu system of the Workbench. Such third-party applications can then be launched via the graphical menu system of CLC Workbenches that are connected to the *CLC Science Server*. These tools can access data on the machine the CLC Workbench is installed on, data stored on the *CLC Science Server* or data stored in areas of the server accessible to the *CLC Science Server*, depending on choices made by the server administrator.

The integration of third party external applications is configured in the *CLC Science Server* administrative web interface. Third party programs configured as External Applications are executed on the machine that the *CLC Science Server* is installed on. Thus, the server administrator has full control over the execution environment.

A special plug-in needs to be installed in the client Workbench to give the end-user access to the configured third-party tools. Please contact support@clcbio.com to get this Workbench plug-in.

An important note about the execution of External Applications: It is important to consider that, like other tasks executed via the *CLC Science Server*, any tool configured as an External Application, will be run as the **same logical user** that runs the *CLC Science Server* process itself. *If you plan to configured External Applications, we highly recommend that you run the CLC Server software as an un-privileged user.* Or, in other words, if your system's root user is running the *CLC Science Server* process, then tasks run via the External Applications functionality will also be executed by the root system user. This would be considered undesirable in most contexts.

Figure 7.1 shows an overview of the actions and data flow that occur when an integrated external applications is executed via the CLC Workbench.

In general terms the basic work flow is:

1. The user selects input data and parameters and starts the job from the Workbench
2. The server exports the input data to a temporary file
3. The server starts the command line application, using the parameters specified from the user and the temporary file as input
4. When the command line application is done, the server imports output data back into the CLC environment, saving it in the data location on the server

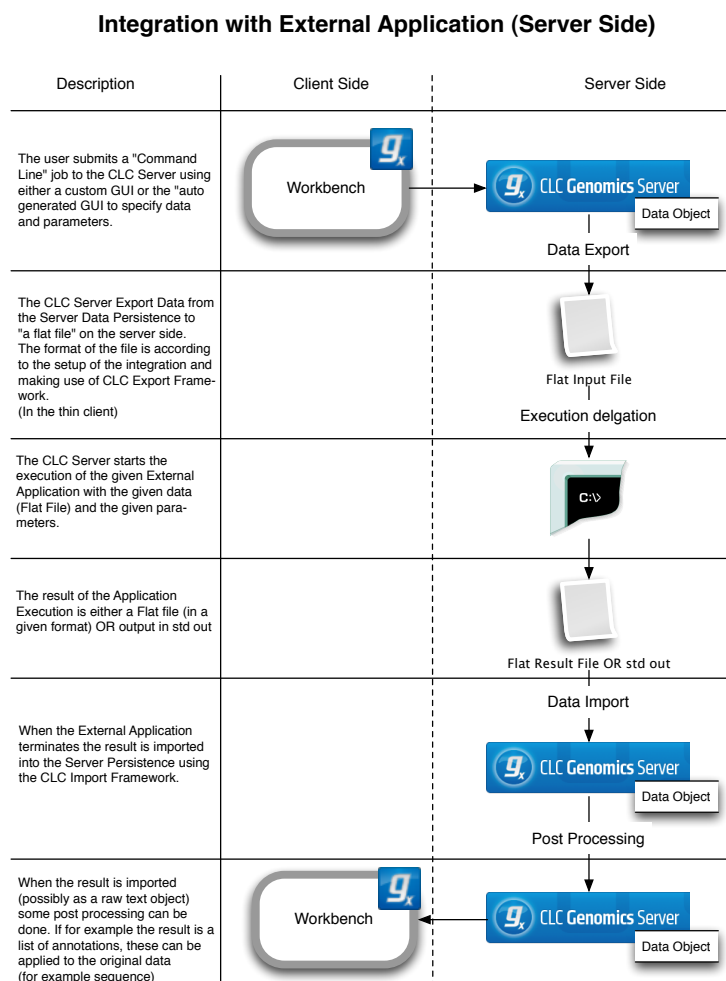


Figure 7.1: An overview of the external applications integration.

5. The user is notified that the job is done, and the results are available for viewing and further analysis in the Workbench

Note that all files used and files created and saved are within the CLC environment. Temporary files are created outside the CLC environment during the execution of a third party tool, but are deleted after the process runs under normal circumstances.

The best way to describe the integration of third party command lines tools is through a series of examples. We start with a very basic example and work towards more complex setups.

7.1 External applications integration: Basic configuration

Many aspects of configuring external tools in the *CLC Science Server* can be described as we set up a very simple command. We have chosen the `cp` command as will already be on your server.

The `cp` command requires at minimum two parameters, an input file and an output file. These parameters are positional: the first filename given after the command is the input file, the second is the output file. Here we will just copy a fasta file from one place in the *CLC Science Server* to another. This is a very inefficient way of doing this task, but it will illustrate how to integrate a

command line tool without requiring you to install additional software on your system.

Under the External Applications tab of the *CLC Science Server* administrative web interface, click on the *New configuration* button. This brings up a window like that shown at the left side of figure 7.2. In the text box labeled *External applications command name*, enter a name for this command. This will be what the end-user sees in the menu option they will be presented with via the Workbench. In the text box labeled *command line argument*, provide the command line. Start with the command, and within curly brackets, include any parameter that needs to be configured by the user. The names of the parameters inside the curly brackets will become the labels of the choices offered to the end-user when they start up this external application via their Workbench. In the right hand side of figure 7.2, we show how this looks if we give the *cp* command two parameters: *infile* and *outfile*.

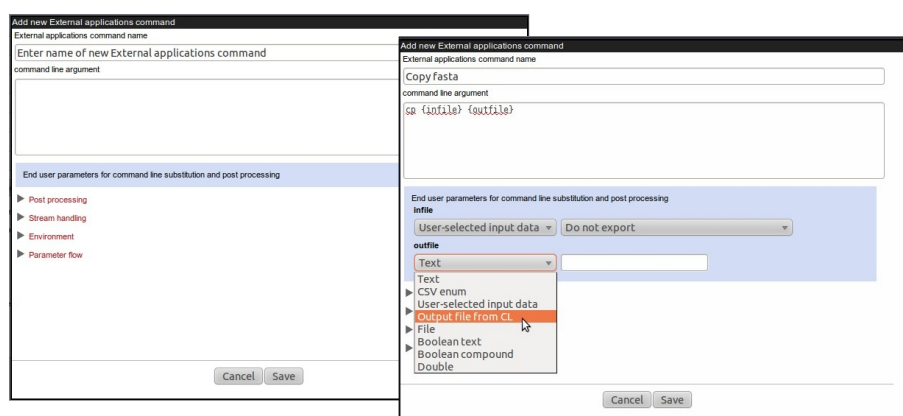


Figure 7.2: Setting up the *cp* command as an external application.

Two drop-down menus have now appeared in the blue shaded area of the right hand window in 7.2. These are dynamically generated. Each parameter you enter in curly brackets in the command text box area will have a drop-down menu created for it. The text you entered within the curly brackets is used to label the entries in the administrative interface, and are also the labels used in the end-user interface presented via the Workbench.

The administrator now chooses the type of data each parameter will refer to. The options are:

- Text - the users is presented with a text box allowing them to enter a parameter value. The administrator can provide a default value if desired.
- CSV enum - this allows the administrator to set a drop down list of parameter choices for the user. For an example of this, please see section 7.6 on setting up Velvet as an external application.
- User-selected input data - users will be prompted to select an input file from those they have stored on the CLC Server.
- Output file from CL - users will be prompted for a location to store a file that is created by the third-party application. An extra text box is also provided in the configuration so the administrator can specify a default name for the re-imported file. If no filename is provided by the administrator, the basename of the file from the system is used.
- File - users can select an input file from their local machine's filesystem.

- Boolean text - This generates a checkbox in the end-user interface, labelled with the text you provide. If the user clicks in the box, the parameter is set to true; an empty box means the parameter is set to false.
- Boolean compound - this enables the creation of a checkbox, where if checked, the end-user is presented with another option (of your choice). If the check box is not checked, then that option will be greyed out. Here, the administrator can also choose if the box is to be checked or unchecked by default in the Workbench interface.
- Double - Allows the user to enter a number. The administrator can choose a number this option should be set to by default. If none is set, then 0 is the default.

In the right hand side of figure 7.2 we set the parameters so that the input file to be sent to the system's copy command will be specified by the user, and we tell the system to export this file from the CLC Server as a fasta file. We then configure the import of output file from the copy command back into the CLC Server, and specify that we are importing a fasta file. When configuring standard bioinformatics third party applications, you are able to choose from many standard formats to export from, and to import back into, the *CLC Science Server*.

Once the configuration is complete and has been saved, the external application should now appear in the list in the administrative web interface.

The small checkbox to the left of the external application name should be checked. This means it is will be accessible to those with the Workbench plug-in installed. If a particular external application needs to be removed from end-user access for a while, this small box can just be unchecked.


7.2 External applications integration: Post-processing

A general description of this part of External Applications is planned. In the meantime, there is a specific example, with an explanation of how it works, within the example for Bowtie integration.

7.3 External applications integration: Stream handling

There is also a general configuration of stream handling available.

The stream handling shown in figure 7.3 allows you to specify where standard out and standard error for the external application should be handled.



The screenshot shows a configuration window titled "Stream handling" with a dropdown arrow. It contains two sections: "Standard out handling:" with a dropdown menu set to "Do not import", and "Standard error handling:" with two radio buttons. The first radio button, "Anything on standard error is shown as user error dialog and execution is stopped", is selected. The second radio button is "Do not stop execution or show error dialogs". Below the radio buttons is another dropdown menu set to "Do not import".

Figure 7.3: Stream handling.

Basically, you can choose to ignore it, or you can import it using one of the importers available on the server. For some applications, standard out produces the main result, so here it makes sense to choose an appropriate importer. But also for debugging purposes it can be beneficial to import standard out and standard error as text so that you can see it in the Workbench after a run.

7.4 External applications integration: Environment

7.4.1 Environmental Variables

This section is still being written.

7.4.2 Working directory

Define the area where temporary files will be stored. The *Default temp-dir* option uses the directory specified by the `java.io.tmpdir` setting for your system. The *Shared temp-dir* option allows you to set one of the directories you have already specified as an *Import/export directory* as the area to be used for temporary files created.

Choosing the Shared temp-dir option means that temporary files created will be accessible to all execution nodes and the master server, without having to move them between machines. In contrast, for a setup with CLC execution nodes that chooses the Default temp-dir setting, where such a directory is usually not shared between machines, files will be moved between the master and job node. The Default temp-dir setting will not work for any setup where the CLC Grid Worker will be used to submit jobs to a local grid.

If you work on a single server, then the Shared temp-dir setting can be used to specify a non-default area for temporary files.

If you work on a master-execution node setup, whether it be grid nodes or CLC execution nodes, the *Shared temp-dir* must be chosen, and this area must:

- Be configured in the Import/Export directories area under the Main Configuration tab
- Be a shared directory, accessible to your master server and all execution nodes

7.4.3 Execute as master process

The checkbox to **Execute as master process** can be checked if the process does not involve intensive processing on the server side. Setting this means that the process will always be run on the Master server. For a single server setup, this has no effect. However, in a system with execution nodes, checking this option results in the queue being effectively by-passed, as the job will be run directly on the master server. This choice will usually only make sense for tasks that require little RAM or cpu. Jobs run this way should will not actually block the queue - they just run as a master process.

7.5 External applications integration: Parameter flow

To aid in determining how the various parameters configured flow through the External Applications process, you can access small graphs showing how the parameters entered in the user

parameters section will be used. You can see an example for the very simple copy command used as an example above in figure 7.4.

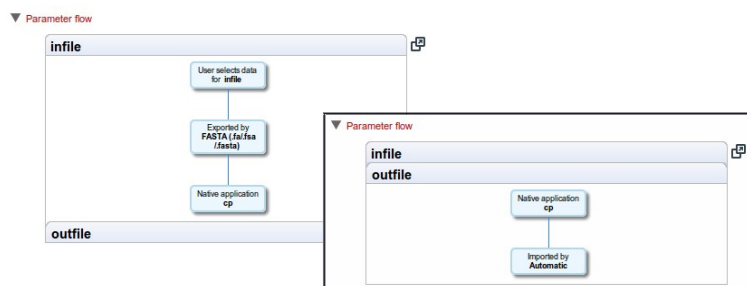


Figure 7.4: An example of the parameter overview facility.

7.6 External applications integration: Velvet

Velvet [Zerbino and Birney, 2008] is a popular de novo assembler for next-generation sequencing data. We have provided example scripts and configurations to set this up as an external application on CLC Science Server.

The velvet package includes two programs that need to be run consecutively. Because the external application on the CLC Science Server is designed to call one program, a script is needed to encapsulate this.

7.6.1 Installing Velvet

To get started, you need to do the following:

- Install Velvet on the server computer (download from <http://www.ebi.ac.uk/~zerbino/velvet/>). Note that if you have job nodes, it needs to be installed on all nodes that will be configured to run Velvet. We assume that Velvet is installed in `/usr/local/velvet` but you can just update the paths if it is placed elsewhere.
- Download the scripts and configuration files made by CLC bio from <http://www.clcbio.com/external-applications/velvet.zip>
- Unzip the file and place the `clcbio` folder and contents in the Velvet installation directory (this is the script tying the two Velvet program together). You need to edit the script if you did not place the Velvet binary files in `/usr/local/velvet`.
- Make sure execute permissions are set on the script and the executable files in the Velvet installation directory. Note that the user executing the files will be the user who started the Server process (if you are using the default start-up script, this will be `root`).
- Use the `velvet.xml` file as a new configuration on the server: Log into the server via the web interface and go to the **External applications** (🔧) tab under **Admin** (👤) and click **Import Configuration**.

When the configuration has been imported, click the **CLC bio Velvet** header and you should see a configuration as shown in figure 7.5.

▼ CLC bio Velvet

External applications command name
CLC bio Velvet

command line argument
`/opt/local/velvet/gclcbio/velvet.sh {hash size} {read type} {reads} {expected coverage} {contigs}`

End user parameters for command line substitution and post processing

hash size
Double ▾ 31

read type
CSV enum ▾ -short -shortPaired -long Short, Short Paired, Long

reads
User-selected input data ▾ FASTA (.fa/.fsa/.fasta) ▾

expected coverage
Double ▾ 10

contigs
Output file from CL ▾ FASTA (.fa/.fsa/.fasta) ▾ contigs

Figure 7.5: The Velvet configuration has been imported.

Update the path to the Velvet installation at the very top if necessary.

7.6.2 Running Velvet from the Workbench

Next step is to test if it can actually be executed. Open the Workbench with the **External Applications Client Plug-in** installed. Go to:

Toolbox | CLC Server (5) | External Applications (▶)

You will now see a list of all the applications that have been set up (figure 7.6).

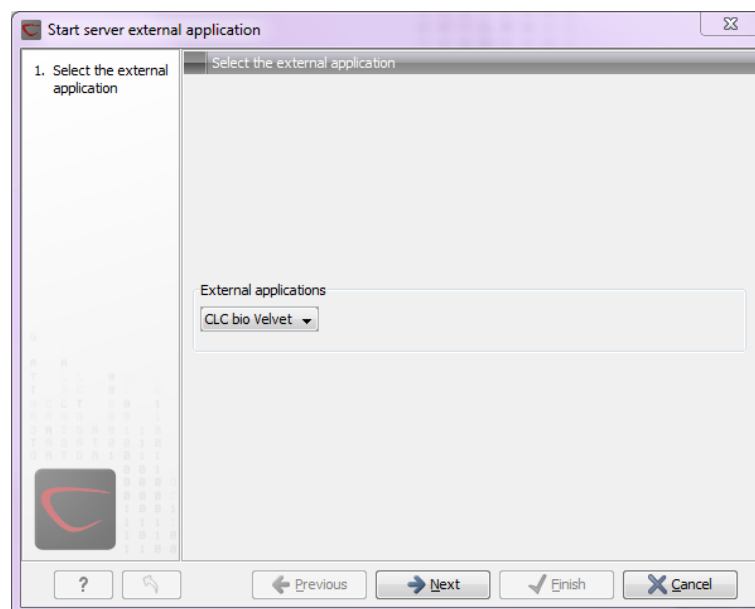


Figure 7.6: Running Velvet from the Workbench.

In this case there is only one. When you click **Next**, you can select (📁) some sequences, set a few parameters and click **Next** and **Finish**.

The process that follows has four steps:

1. The sequencing reads are exported by the server to a fasta file. The fasta file is a temporary file that will be deleted when the process is done.
2. The velvet script is executed using this fasta file and the user-specified parameters as input.
3. The resulting output file is imported into the save location specified in the save step of the Workbench dialog, and the user is notified that the process is done.
4. All temporary files are deleted

7.6.3 Understanding the Velvet configuration

We will now explain how the configuration that we made actually works. And hopefully this will make it possible for you to design your own integrations.

Going back to figure 7.5, there is a text field at the top. This is where the command expression is created, in this case:

```
/opt/local/velvet/clcbio/velvet.sh {hash size} {read type}  
{reads} {expected coverage} {contigs}
```

The first is the path to the script, and the following are parameters that are interpreted by the server when calling the script because they are surrounded by curly brackets `{ }`. Note that each parameter entered in curly brackets gets an entry in the panel below the command line expression.

The first one, `hash size`, can be entered as a **Double** (which is a number in computer parlance) and it is thus up to the user to provide a value. A default value is entered here in the configuration (31).

The second one is the `read type` which has been configured as a **CSV enum** which is basically a list. The first part consists of the parameters to be used when calling the script (`-short`, `-shortPaired`, `-long`, `-longPaired`), and the second part is the more human-readable representation that is shown in the Workbench (`Short`, `Short Paired`, `Long`, `Long Paired`).

The third parameter is `reads` which is the input data. When the **User-selected input data** option is chosen, a list of all the available export formats is presented. In this case, Velvet expects a fasta file. When a user starts Velvet from the Workbench, the server starts exporting the selected input data to a temporary fasta file before running the script.

The `expected coverage` is similar to `hash size`.

The last parameter is `contigs` which represents the output file. This time, a list of import data formats is available used to import the data back into the folder that the user selected as save destination.

7.7 Troubleshooting

7.7.1 Checking the configuration

Since there is no check of consistency of the configuration when it has been set up, errors will only be seen on runtime when the application is executed. In order to help trouble-shooting in case of problems, there are a few things that can be done:

First, in the error dialog that will be presented in the workbench, you can see the actual command line call in the **Advanced** tab at the bottom. This can be a great help identifying syntax errors in the call.

Second, if you choose to import standard out and standard error as text, this will make it possible to check error messages posted by the external application (see figure 7.7).

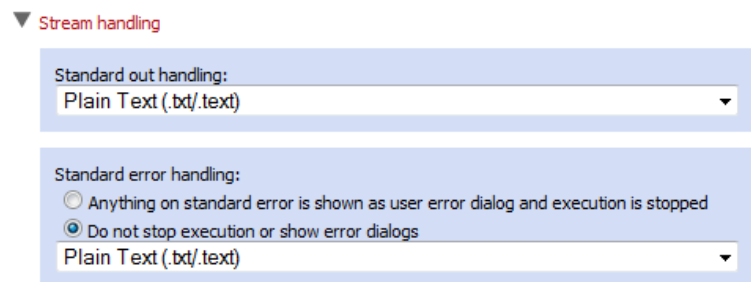


Figure 7.7: Importing the result of standard error and standard out.

Once the set-up is running and stable, you can deselect these options.

7.7.2 Check your third party application

- Is your third party application being found? Perhaps try giving the full path to it.
- If you are using someone else's configuration file, make sure the location to the third party application is correct for your system.
- If you are using someone else's wrapper scripts, make sure all locations referred to inside the script are correct for your system.
- Is your third party application executable?
- If there was a wrapper script being used to call the third party application, is that wrapper script executable?

7.7.3 Is your Import/Export directory configured?

For certain setups, you need to have Import/Export directories configured. Please refer to section 7.4.2 for more details on this.

7.7.4 Check your naming

If your users will only access the External Applications via the Workbench, then you do not have to worry about what name you choose when setting up the configuration. However, if they plan to

use the **clcserver** program, from the CLC Command Line Tools, to interact with your *CLC Science Server*, then please ensure that you do not use the same name as any of the internal commands available. You can get a list of these by running the `clcserver` command, with your *CLC Science Server* details, and using the `-A` flag with no argument.

Chapter 8

Workflows

The *CLC Science Server* supports workflows that are created with the CLC Workbenches. A workflow consists of a series of tools where the output of one tool is connected as the input to another tool. As an example, a workflow could pass data through read mapping, use the mapped reads as input for variant detection, and perform some filtering of the variant track. The workflow is created in the CLC Workbench and an installer file is created that can be installed on the *CLC Science Server*. For information about creating a workflow, please see the user manual of *CLC Genomics Workbench* or *CLC Main Workbench* at <http://www.clcbio.com/usermanuals>.

8.1 Installing and configuring workflows

Workflows can be installed from the server web interface:

Admin (⚙️) | **Workflows** (📁)

Click the **Install Workflow** button and select a workflow installer (for information about creating a workflow, please see the user manual of *CLC Genomics Workbench* or *CLC Main Workbench* at <http://www.clcbio.com/usermanuals>).

Once installed, the workflow is listed with a validated (✅) or attention (⚠️) status icon as shown in figure 8.1).

In this example, there are several workflow elements that can be configured. Simply click the box and you will see a dialog listing the parameters that need to be configured as well as an overview of all the parameters. An example is shown in figure 8.2.

In addition to the configuration of values for the open parameters, you can also specify which of those open parameters that should be locked (this means that the parameter cannot be changed when executing the workflow). Learn more about locking and unlocking parameters in the user manual of *CLC Genomics Workbench* or *CLC Main Workbench* at <http://www.clcbio.com/usermanuals>.

8.2 Executing workflows

Once a workflow is installed and validated, it becomes available for execution. When you log in on the server using the CLC Workbench, workflows installed on the server automatically become

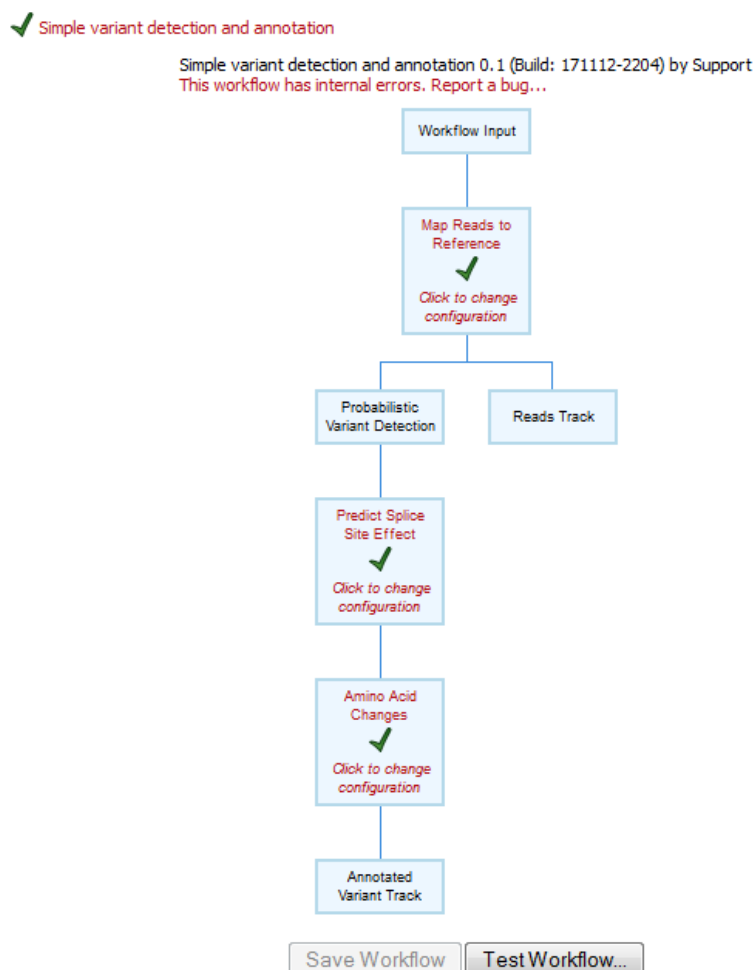


Figure 8.1: A workflow is installed and validates.

available in the **Toolbox** (see figure 8.3).

When you select it, you will be presented with a dialog as shown in figure 8.4 with the options of where to run the workflow.

This means that workflows installed on the server can be executed either on the server or in the workbench. In the same way, workflows installed on the workbench can be executed on the server as well as on the workbench. The only requirement is that both the tools that are part of the workflow and any reference data are available.

An important benefit of installing workflows in the server is that it provides the administrator an easy way to update and deploy new versions of the workflow, because any changes immediately take effect for all workbench users as well.

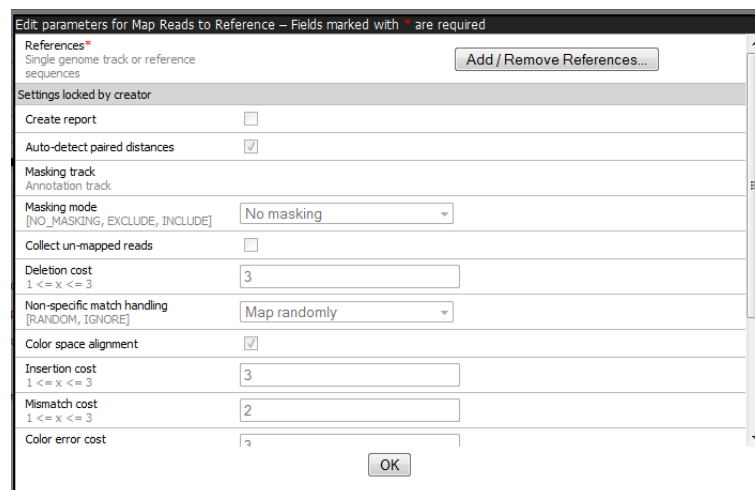


Figure 8.2: The reference sequence is the only parameter that can be configured for the read mapping.

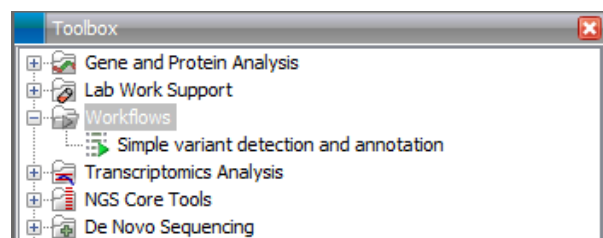


Figure 8.3: A workflow is installed and ready to be used.

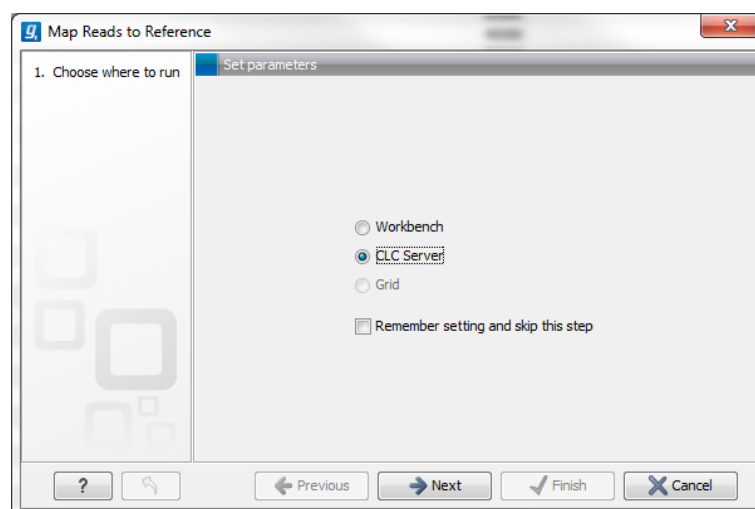


Figure 8.4: Selecting where to run the workflow.

Chapter 9

Appendix

9.1 Troubleshooting

If there are problems regarding the installation and configuration of the server, please contact support@clcbio.com.

9.1.1 Check set-up

In order to check that your server has been set up correctly, you can run the **Check set-up** tool. Log in on the web interface of the server as an administrator and click the **Check Set-up** link at the upper right corner. This will show a dialog where you click **Generate Diagnostics Report**.

This will show a list of test that are performed on the system as shown in figure 9.1.

If any of the tests fail, it will be shown in the list. You can expand each of the tests to display more information about what the test is checking and information about the error if it fails.

9.1.2 Bug reporting

When contacting support@clcbio.com regarding problems on the server, you will often be asked for additional information about the server set-up etc. In this case, you can easily send the necessary information by submitting a bug report:

Log in to the web interface of the server as administrator | report a bug (at the top right corner) | Enter relevant information with as much detail as possible | Submit Bug Report to CLC bio

You can see the bug report dialog in 9.2.

The bug report includes the following information:

- Log files
- A subset of the audit log showing the last events that happened on the server
- Configuration files of the server configuration

In a job node set-up you can include all this information from the job nodes as well by checking the **Include comprehensive job node info** checkbox in the **Advanced** part of the dialog.



Figure 9.1: Check system. Failed elements will be marked with a red X. If you have not configured your Server to submit jobs to a local Grid system, or if you have and your setup is configured correctly, you will see a green checkmark beside the Grid setup status item in the diagnostic report.

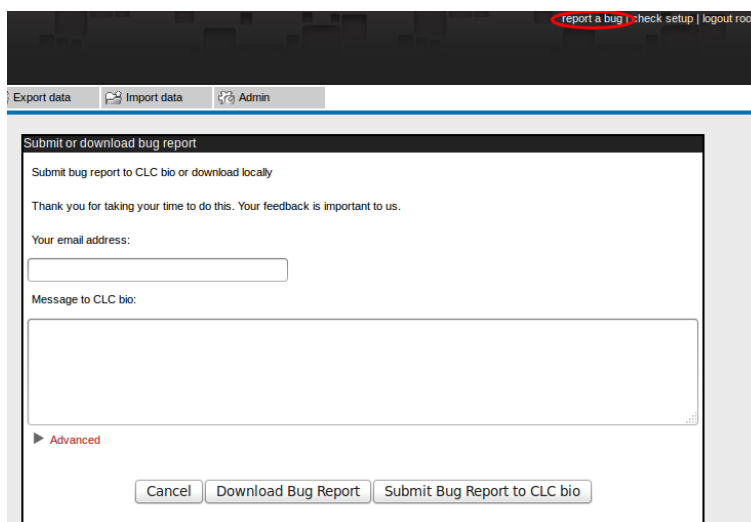


Figure 9.2: Submitting a bug report to CLC bio.

If the server does not have access to the internet, you can **Download bug report**. This will create a zip file containing all the information and you can pass that on to CLC bio support. If the server has access to the internet, you can **Submit Bug Report to CLC bio**.

Note that the process of gathering the information for the bug report can take a while, especially for job node set-ups. If a Workbench user experiences a server-related error, it is also possible to submit a bug report from the Workbench error dialog. This report will include the same archive as when submitting a bug report from the web interface. All data sent to support@clcbio.com is treated confidentially.

No password information is included in the bug report.

9.2 Database configurations

9.2.1 Configurations for MySQL

For MySQL we recommend basing your configuration on the example configuration file `my-large.cnf` which is included in the MySQL distribution.

In addition the following changes should be made:

The `max_allowed_packet` should be increased to allow transferring large binary objects to and from the database. This is done by setting the option: `max_allowed_packet = 64M`

InnoDB must be available and configured for the MySQL instance to work properly as the CLC Database. You should enable the options in the InnoDB section of your configuration as suggested below:

```
# You can set .._buffer_pool_size up to 50 - 80 %
# of RAM but beware of setting memory usage too high
innodb_buffer_pool_size = 256M
innodb_additional_mem_pool_size = 20M
# Set .._log_file_size to 25 % of buffer pool size
innodb_log_file_size = 64M
innodb_log_buffer_size = 8M
innodb_flush_log_at_trx_commit = 1
innodb_lock_wait_timeout = 50
```

There appears to be a bug in certain versions of MySQL which can cause the cleanup of the query cache to take a very long time (some time many hours). If you experience this you should disable the query log by setting the following option: `query_cache_size= 0`

9.3 SSL and encryption

The *CLC Science Server* supports SSL communication between the server and its clients (e.g. Workbenches or the *CLC Server Command Line Tools*). This is particularly relevant if the server is accessible over the internet as well as on a local network.

The default configuration of the server does not use SSL.

9.3.1 Enabling SSL on the server

A **server certificate** is required before SSL can be enabled on the *CLC Science Server*. This is usually obtained from a *Certificate Authority* (CA) like Thawte or Verisign (see http://en.wikipedia.org/wiki/Certificate_authorities).

A **signed certificate** in a `pcks12` keystore file is also needed. The keystore file is either provided by the CA or it can be generated from the private key used to request the certificate and the signed-certificate file from the CA (see section 9.3.1).

Copy the keystore file to the `conf` subdirectory of the *CLC Science Server* installation folder.

Next, the `server.xml` file in the `conf` subdirectory of the *CLC Science Server* installation folder has to be edited to enable SSL-connections. Add text like the following text to the `server.xml` file:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
    maxThreads="150" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS"
    keystoreFile="conf/keystore.pkcs12" keystorePass="tomcat"
    keystoreType="PKCS12"
/>
```

Replace `keystore.pkcs12` with the name of your keystore file, and replace `tomcat` with the password for your keystore.

The above settings make SSL available on port 8443. The standard (non-SSL) port would still be 7777, or whatever you may have configured it to. If only SSL connection should be allowed, the connector entry for port 7777 can safely be removed from the `server.xml` file.

Self-signed certificates can be generated if only connection encryption is needed. See http://www.akadia.com/services/ssh_test_certificate.html for further details.

Creating a PKCS12 keystore file

If the certificate is not supplied in a `pkcs12` keystore file, it can be put into one by combining the private key and the signed certificate obtained from the CA by using `openssl`:

```
openssl pkcs12 -export -out keystore.pkcs12 -inkey private.key -in certificate.crt -name "tomcat"
```

This will take the private key from the file `private.key` and the signed certificate from `certificate.crt` and generate a `pkcs12`-store in the `keystore.pkcs12` file.

9.3.2 Logging in using SSL from the Workbench

When the Workbench connects to the *CLC Science Server* it automatically detects if Secure Socket Layer (SSL) should be used or not.

If SSL is detected, the server's certificate will be verified and a warning is displayed if the certificate is not signed by a recognized Certificate Authority (CA) as shown in figure 9.3.

When such an "unknown" certificate has been accepted once, the warning will not appear again. It is necessary to log in again once the certificate has been accepted.

When logged into a server, information about the connection can be viewed by hovering the connection icon on the status-panel as shown in figure 9.4.

The icon is gray when the user is not logged in, and a pad lock is overlayed when the connection is encrypted via SSL.

9.3.3 Enabling redirection from non-ssl port to ssl-enabled port

If SSL is to be mandatory, it is possible to get the non-ssl port (7777) to forward to the ssl port (8443), rather than closing down the non-ssl port. This is done by adding the following section to the `conf/web.xml` file:

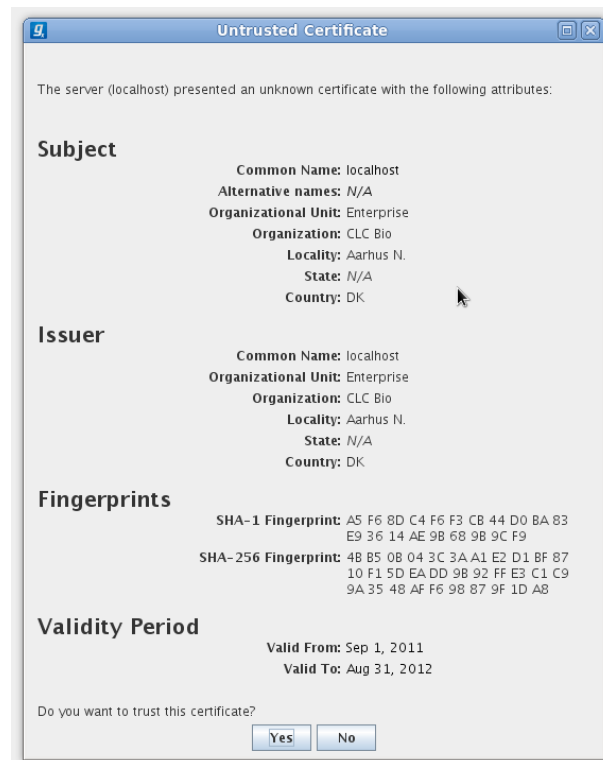


Figure 9.3: A warning is shown when the certificate is not signed by a recognized CA.

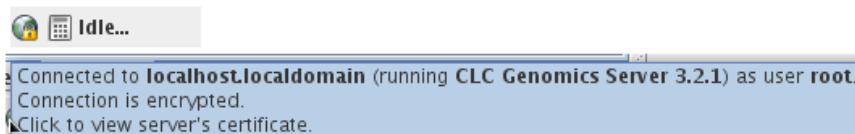


Figure 9.4: Showing details on the server connection by placing the mouse on the globe.

```
<security-constraint>
<display-name>Security Constraint</display-name>
<web-resource-collection>
  <web-resource-name>SSL-restricted Area</web-resource-name>
  <url-pattern>/*</url-pattern>
</web-resource-collection>
<user-data-constraint>
  <transport-guarantee>CONFIDENTIAL</transport-guarantee>
</user-data-constraint>
</security-constraint>
```

The above section is commented out in the supplied `conf/web.xml` file, and just has to be commented in in order for redirection to work. Please note that this redirection only works for the browser accessing the web interface. When connecting from the Workbench, the correct port has to be specified.

9.3.4 Logging in using SSL from the CLC Server Command Line Tools

The CLC Server Command Line Tools will also automatically detect and use SSL if present on the port it connects to. If the certificate is untrusted the `clcserver` program will refuse to login:


```
./clserver -S localhost -U root -W defaultt -P 7778
Message: Trying to log into server
Error: SSL Handshake failed. Check certificate.
Option          Description
-----
-A <Command>    Command to run. If not specified the list of commands on the server will be returned.
-C <Integer>    Specify column width of help output.
-D <Boolean>    Enable debug mode (default: false)
-G <Grid Preset value> Specify to execute on grid.
-H             Display general help.
-I <Algorithm Command> Get information about an algorithm
-O <File>       Output file.
-P <Integer>    Server port number. (default: 7777)
-Q <Boolean>    Quiet mode. No progress output. (default: false)
-S <String>    Server hostname or IP-address of the CLC Server.
-U <String>    Valid username for logging on to the CLC Server
-V            Display version.
-W <String>    Clear text password or domain specific password token.
```

In order to trust the certificate the `sslStore` tool must be used:

```
./sslStore -S localhost -U root -W defaultt -P 7778
The server (localhost) presented an untrusted certificate with the following attributes:
SUBJECT
=====
Common Name       : localhost
Alternative Names : N/A
Organizational Unit: Enterprise
Organization      : CLC Bio
Locality          : Aarhus N.
State             : N/A
Country           : DK

ISSUER
=====
Common Name       : localhost
Organizational Unit: Enterprise
Organization      : CLC Bio
Locality          : Aarhus N.
State             : N/A
Country           : DK

FINGERPRINTS
=====
SHA-1             : A5 F6 8D C4 F6 F3 CB 44 D0 BA 83 E9 36 14 AE 9B 68 9B 9C F9
SHA-256           : 4B B5 0B 04 3C 3A A1 E2 D1 BF 87 10 F1 5D EA DD 9B 92 FF E3 C1 C9 9A 35 48 AF F6 98 87 9F 1D A8

VALIDITY PERIOD
=====
Valid From        : Sep 1, 2011
Valid To          : Aug 31, 2012
Trust this certificate? [yn]
```

Once the certificate has been accepted, the `clserver` program is allowed to connect to the server.

Bibliography

[Zerbino and Birney, 2008] Zerbino, D. R. and Birney, E. (2008). Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res*, 18(5):821–829.

Index

Active directory, [31](#)
AD, [31](#)
Automation, [65](#)

Bibliography, [74](#)

Command-line installation, [15](#)
Cores, restrict usage, [27](#)
CPU, restrict usage of, [27](#)

Encrypted connection, [70](#)
External applications, [55](#)

GSSAPI, [31](#)

HTTPS, [70](#)

Kerberos, [31](#)

LDAP, [31](#)

Memory allocation, [26](#)

permissions, [36](#)
Pipeline, [65](#)

Quiet installation, [15](#)

RAM, [26](#)
Recycle bin, [38](#)
References, [74](#)

Scripting, [65](#)
Secure socket layer, [70](#)
Silent installation, [15](#)
SSL, [70](#)
System requirements, [8](#)

tmp directory, how to specify, [26](#)

 .vmoptions, memory allocation, [26](#)

Workflow, [65](#)

Xmx argument, [26](#)