GxS

BxS

CLC **Server**

Administrator

USER MANUAL

Administrator Manual for

CLC Server 9.0

Windows, Mac OS X and Linux

February 28, 2017

This software is for research purposes only.

QIAGEN Aarhus
Silkeborgvej 2
Prismet
DK-8000 Aarhus C
Denmark

Contents

Chapter 1

Introduction

Welcome to *CLC Server 9.0*, a central element of the CLC product line enterprise solutions.

The latest version of the user manual can also be found in pdf format at `http://www.qiagenbioinformatics.com/support/manuals/`.

You can get an overview of the server solution in figure 1.1. The software depicted here is for research purposes only.

Using a server means that data can be stored centrally and analyses run on a central machine rather than a personal computer. Please see section 1.3 and section 1.4 for a listing of tools shipped with CLC Servers.

After logging into the CLC Server from a Workbench, data on the server will be listed in the Workbench navigation area and analyses can be started as usual. The key difference is that when you are logged into a CLC Server from a Workbench, you will be get the choice of where to run the analysis: on the Workbench or on the CLC Server.

1.1 System requirements

The system requirements of *CLC Server* are:

Server operating system

- Windows 7, Windows 8, Windows 10 or Windows Server 2012
- Mac OS X 10.9, 10.10, 10.11 and 10.12
- Linux: RHEL 6.0 and later, SUSE 13.1 and later
- 64 bit
- For CLC Server setups that include job nodes and grid nodes, those nodes must run the same type of operating system as the master CLC Server

Server hardware requirements

- Intel or AMD CPU required

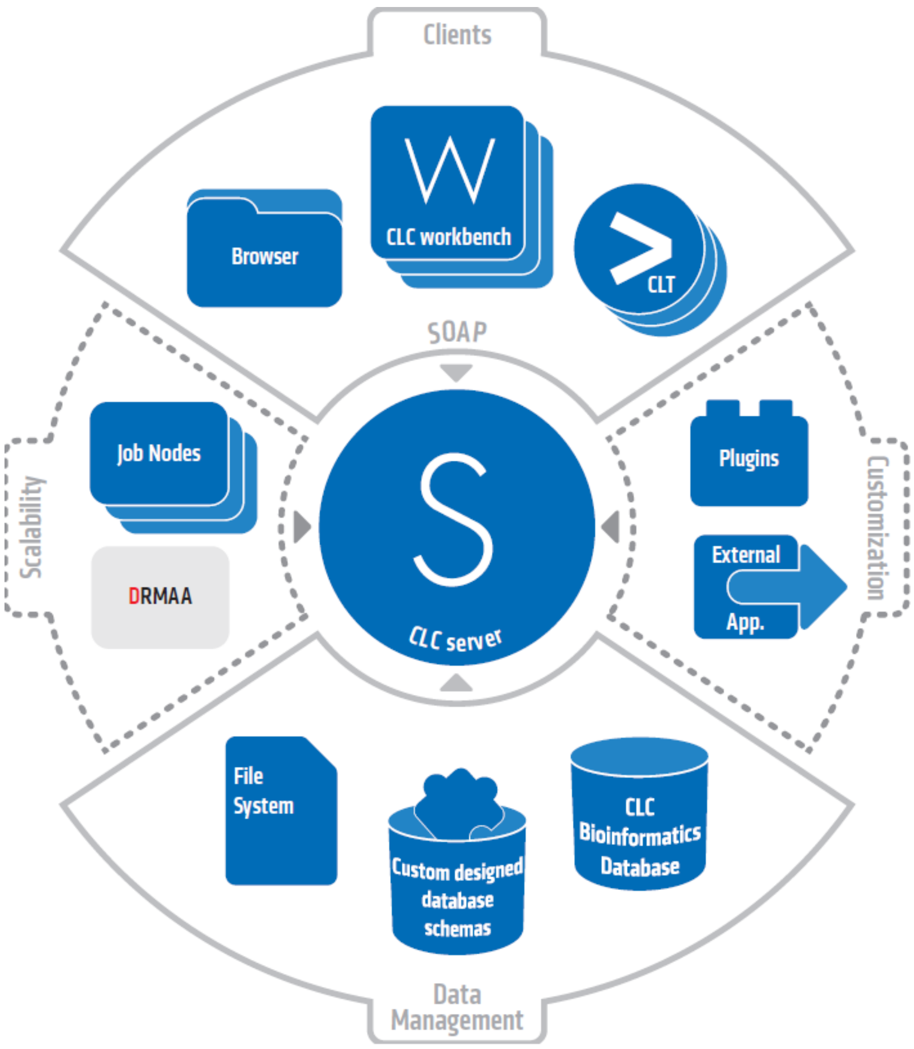


Figure 1.1: *An overview of the server solution. Note that not all features are included with all license models.*

- Computer power: 2 cores required. 8 cores recommended.
- Memory: CLC Genomics Server: 4 GB RAM required, 16 GB RAM recommended. Biomedical Genomics Server Extension requires 16 GB RAM at least.
- Disk space: 500 GB required. More needed if large amounts of data are analyzed.

Special memory requirements for working with genomes

The numbers below give minimum and recommended amounts for systems running mapping and analysis tasks. The requirements suggested are based on the genome size.

- **E. coli K12 (4.6 megabases)**
  - Minimum: 2 GB RAM
  - Recommended: 4 GB RAM
- **C. elegans (100 megabases)** and **Arabidopsis thaliana (120 megabases)**

  - Minimum: 2 GB RAM
  - Recommended: 4 GB RAM

- **Zebrafish (1.5 gigabases)**
  - Minimum: 2 GB RAM
  - Recommended: 4 GB RAM

- **Human (3.2 gigabases)** and **Mouse (2.7 gigabases)**
  - Minimum: 6 GB RAM
  - Recommended: 8 GB RAM

Special requirements for de novo assembly

De novo assembly may need more memory than stated above - this depends both on the number of reads and the complexity and size of the genome. See http://resources.qiagenbioinformatics.com//white-papers/White_paper_on_de_novo_assembly_4.pdf for examples of the memory usage of various data sets.

1.2 Licensing

Three kinds of license can be involved in running analyses on the *CLC Server*.

- **A license for the server software itself**. This is needed for running analyses via the server. The license will allow a certain number of open sessions. This refers to the number of active, individual log-ins from server clients such as Workbenches, the Command Line Tools, or the web interface to the server. The number of sessions is part of the agreement with QIAGEN when you purchase a license. The manual chapter about installation provides information about how to obtain and deploy the license for the server.
- **A license for the Workbench software.** The Workbench is used to launch analyses on the server and to view the results. Find the user manuals and deployment manual for the Workbenches at http://www.qiagenbioinformatics.com/support/manuals/.
- **A network license if you will be submitting analyses to grid nodes.** This is explained in detail in section 6.3.5.

1.3 CLC Genomics Server

The *CLC Genomics Server* is shipped with the following tools and analyses that can all be started from *CLC Genomics Workbench* and *CLC Server Command Line Tools*:

- Import
- Export
- Download Reference Genome Data
- Search for Reads in SRA

- Classical Sequence Analysis
  - Create Alignment (Alignments and Trees)
  - K-mer Based Tree Construction (Alignments and Trees)
  - Create Tree (Alignments and Trees)
  - Model Testing (Alignments and Trees)
  - Maximum Likelihood Phylogeny (Alignments and Trees)
  - Extract Annotations (General Sequence Analysis)
  - Extract Sequences (General Sequence Analysis)
  - Motif Search (General Sequence Analysis)
  - Translate to Protein (Nucleotide Analysis)
  - Convert DNA to RNA (Nucleotide Analysis)
  - Convert RNA to DNA (Nucleotide Analysis)
  - Reverse Complement Sequence (Nucleotide Analysis)
  - Reverse Sequence (Nucleotide Analysis)
  - Find Open Reading Frames (Nucleotide Analysis)
  - Download Pfam Database (Protein Analysis)
  - Pfam Domain Search (Protein Analysis)
- Molecular Biology Tools
  - Assemble Sequences (Sequencing Data Analysis)
  - Assemble Sequences to Reference (Sequencing Data Analysis)
  - Secondary Peak Calling (Sequencing Data Analysis)
  - Find Binding Sites and Create Fragments (Primers and Probes)
  - Add attB Sites (Cloning and Restriction Sites - Gateway Cloning)
  - Create Entry clone (BP) (Cloning and Restriction Sites - Gateway Cloning)
  - Create Expression clone (LR) (Cloning and Restriction Sites - Gateway Cloning)
- BLAST
  - BLAST
  - BLAST at NCBI
  - Download BLAST Databases
  - Create BLAST Database
- NGS Core Tools
  - Sample Reads
  - Create Sequencing QC Report
  - Merge Overlapping Pairs
  - Trim Sequences
  - Demultiplex Reads

  - Map Reads to Reference
  - Local Realignment
  - Create Detailed Mapping Report
  - Merge Read Mappings
  - Extract Consensus Sequence
- Track Tools
  - Convert to Tracks
  - Convert from Tracks
  - Merge Annotation Tracks
  - Annotate with Overlap Information (Annotate and Filter)
  - Extract Reads Based on Overlap (Annotate and Filter)
  - Filter Annotations on Name (Annotate and Filter)
  - Filter Based on Overlap (Annotate and Filter)
  - Create GC Content Graph Tracks (Graphs)
  - Create Mapping Graph Tracks (Graphs)
  - Identify Graph Threshold Areas(Graphs)
- Resequencing Analysis
  - Create Statistics for Target Regions
  - Identify Known Mutations from Sample Mappings
  - InDels and Structural Variants
  - Coverage Analysis
  - Basic Variant Detection (Variant Detectors)
  - Fixed Ploidy Variant Detection (Variant Detectors)
  - Low Frequency Variant Detection (Variant Detectors)
  - Annotate from Known Variants (Annotate and Filter Variants)
  - Filter against Known Variants (Annotate and Filter Variants)
  - Identify Candidate Variants
  - Annotate with Exon Numbers (Annotate and Filter Variants)
  - Annotate with Flanking Sequences (Annotate and Filter Variants)
  - Filter Marginal Variant Calls (Annotate and Filter Variants)
  - Filter Reference Variants (Annotate and Filter Variants)
  - Compare Sample Variant Tracks (Compare Variants)
  - Compare Variants within Group (Compare Variants)
  - Fisher Exact Test (Compare Variants)
  - Trio Analysis (Compare Variants)
  - Filter against Control Reads (Compare Variants)
  - GO Enrichment Analysis (Functional Consequences)

  - Amino Acid Changes (Functional Consequences)
  - Annotate with Conservation Score (Functional Consequences)
  - Predict Splice Site Effect (Functional Consequences)
  - Link Variants to 3D Protein Structure (Functional Consequences)
  - Download 3D Protein Structure Database (Functional Consequences)
- RNA-Seq Analysis
  - RNA-Seq Analysis (RNA-Seq Analysis)
  - PCA for RNA-Seq
  - Differential Expression for RNA-Seq
  - Create Heat Map for RNA-Seq
  - Create Expression Browser
  - Create Venn Diagram for RNA-Seq
  - Gene Set Test
  - Generate combined RNA-Seq Report
- Microarray and Small RNA Analysis
  - Create Track from Experiment
  - Extract and Count (Small RNA Analysis)
  - Annotate and Merge Counts (Small RNA Analysis)
  - Create Box Plot (Quality Control)
  - Hierarchical Clustering of Samples (Quality Control)
  - Principal Component Analysis (Quality Control)
  - Empirical Analysis of DGE (Statistical Analysis)
  - Proportion-based Statistical Analysis (Statistical Analysis)
  - Gaussian Statistical Analysis (Statistical Analysis)
  - Create MA Plot (General Plots)
  - Create Scatter Plot (General Plots)
  - Histogram (General Plots)
- Epigenomics Analysis
  - Transcription Factor ChIP-Seq
  - Annotate with Nearby Gene Information
- De Novo Sequencing
  - De Novo Assembly
  - Map Reads to Contigs

The functionality of the *CLC Genomics Server* can be extended by installation of Server plugins. The available plugins can be found at http://www.qiagenbioinformatics.com/plugins/.

Latest improvements

CLC Genomics Server is under constant development and improvement. A detailed list that includes a description of new features, improvements, bugfixes, and changes for the current version of *CLC Genomics Server* can be found at:

http://www.qiagenbioinformatics.com/products/clc-genomics-server/latest-improvements/current-line/.

1.4 Biomedical Genomics Server Extension

A *CLC Genomics Server* with the *Biomedical Genomics Server Extension* is shipped with the tools and analyses listed below. These can all be started from the *Biomedical Genomics Workbench* or by using the *CLC Server Command Line Tools*.

- Import
- Export
- Download Reference Genome Data
- Search for Reads in SRA
- Genome Browser
  - Create GC Content Graph (Graphs)
  - Create Mapping Graph (Graphs)
  - Identify Graph Threshold Area (Graphs)
- Quality Control
  - QC for Sequencing Reads
  - QC for Target Sequencing
  - QC for Read Mapping
- Preparing Raw Data
  - Merge Overlapping Pairs
  - Trim Sequences
  - Demultiplex reads
- Resequencing Analysis
  - Identify Known Mutations from Sample Mappings
  - Extract Reads Based on Overlap
  - Map Reads to Reference
  - Local Realignment

- Merge Read Mappings
  - Copy Number Variant Detection
  - Remove Duplicate Mapped Reads
  - Indels and Structural Variants
  - Whole Genome Coverage Analysis
  - Basic Variant Detection (Variant Detectors)
  - Fixed Ploidy Variant Detection (Variant Detectors)
  - Low Frequency Variant Detection (Variant Detectors)

- Add Information to Variants
  - Add Information from Variant Databases
  - Add Conservation Scores
  - Add Exon Number
  - Add Flanking Sequence
  - Add Fold Changes
  - Add information about Amino Acids Changes
  - Add Information from Genomic Regions
  - Add Information from Overlapping Genes
  - Link Variants to 3D Protein Structure
  - Download 3D Protein Structure Database
  - Add Information from 1000 Genomes Project (From Databases)
  - Add Information from COSMIC (From Databases)
  - Add Information from Clinvar (From Databases)
  - Add Information from Common dbSNP (From Databases)
  - Add Information from Hapmap (From Databases)
  - Add Information from dbSNP (From Databases)

- Remove Variants
  - Remove Variants Found in External Databases
  - Remove Variants Not Found in External Databases
  - Remove False Positive
  - Remove Germline Variants
  - Remove Reference Variants
  - Remove Variants Inside Genome Regions
  - Remove Variants Outside Genome Regions
  - Remove Variants Outside Targeted Regions
  - Remove Variants Found in 1000 Genomes Project (From Databases)
  - Remove Variants Found in Common dbSNP (From Databases)
  - Remove Variants Found in Hapmap (From Databases)

- Add Information to Genes
  - Add Information from Overlapping Variants
- Compare Samples
  - Compare Shared Variants Within a Group of Samples
  - Identify Enriched Variants in Case vs Control Group
  - Trio Analysis
- Identify Candidate Variants
  - Identify Candidate Variants
  - Remove Information from Variants
  - Identify Variants with Effect on Splicing
- Identify Candidate Genes
  - Identify Differentially Expressed Gene Groups and Pathways
  - Identify Highly Mutated Gene Groups and Pathways
  - Identify Mutated Genes
  - Select Genes by Name
- RNA-Seq Analysis
  - RNA-Seq Analysis (RNA-Seq Analysis)
  - PCA for RNA-Seq
  - Differential Expression for RNA-Seq
  - Create Heat Map for RNA-Seq
  - Create Expression Browser
  - Create Venn Diagram for RNA-Seq
  - Gene Set Test
  - Generate combined RNA-Seq Report
- Microarray and Small RNA Analysis
  - Create Track from Experiment
  - Extract and Count (Small RNA Analysis)
  - Annotate and Merge Counts (Small RNA Analysis)
  - Create Box Plot (Quality Control)
  - Hierarchical Clustering of Samples (Quality Control)
  - Principal Component Analysis (Quality Control)
  - Empirical Analysis of DGE (Statistical Analysis)
  - Proportion-based Statistical Analysis (Statistical Analysis)
  - Gaussian Statistical Analysis (Statistical Analysis)
  - Create MA Plot (General Plots)

  - Create Scatter Plot (General Plots)
  - Histogram (General Plots)
- Helper Tools
  - Extract Sequences
  - Filter Based on Overlap
- Cloning and Restriction Sites
  - Add attB Sites (Gateway Cloning)
  - Create Entry clone (BP) (Gateway Cloning)
  - Create Expression clone (LR) (Gateway Cloning)
- Sanger Sequencing
  - Assemble Sequences (Sequencing Data Analysis)
  - Assemble Sequences to Reference (Sequencing Data Analysis)
  - Secondary Peak Calling (Sequencing Data Analysis)
  - Find Binding Sites and Create Fragments (Primers and Probes)
- Epigenomics Analysis
  - Transcription Factor ChIP-Seq
  - Annotate with Nearby Gene Information

The functionality of the *CLC Server* can be extended by installation of Server plugins. The available plugins can be found at http://www.qiagenbioinformatics.com/plugins/.

Chapter 2

Installation

2.1 Quick installation guide

The following describes briefly the steps needed to set up *CLC Genomics Server* and *Biomedical Genomics Server Extension* with pointers to more detailed explanation of each step.

If you are looking for how to set up your workbench as a client software, please look at the CLC Server End User manual.

If you are looking for how to set up a *CLC License Server*, instructions can be found in the *CLC License Server* manual.

If you are going to set up execution nodes as well, please read section 6 first.

1. Download and run the server software installer file. When prompted during the installation process, choose to start the server (section 2.2).

2. Run the license download script distributed with the server software. This script can be found in the installation area of the software. (section 2.7). The script will automatically download a license file and place it in the server installation directory under the folder called `licenses`.

3. Restart the server (section 2.8).

4. Ensure the necessary port is open for access by client software for the server. The default port is 7777 .

5. Log into the server web administrative interface using a web browser using the username **root** and password **default** (section 3).

6. Change the root password (section 4.1).

7. Configure the authentication mechanism and optionally set up users and groups (section 4.2).

8. Add data locations (section 3.2).

9. From *within the Workbench*, download and install the Workbench Client plugin. This is needed for the Workbench to be able to contact the server (section 2.9).

10. Check your server setup using the **Check set-up** link in the upper right corner as described in section 15.2.1.

11. Your server should now be ready for use.

2.2 Installing and running the Server

Getting the *CLC 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 Server* is straightforward. However, if you do run into troubles, please refer to the troubleshooting section in Appendix 15.2, which provides tips on how to troubleshoot problems yourself, as well as how to get help.

Note that if you have a *Biomedical Genomics Server Extension*, there are important extra notes to read about installation found in section 2.4.

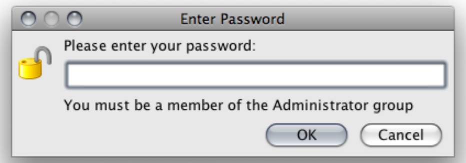
2.2.1 Installing the Server software

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 2.1) or asked to allow the installation to be performed.

- On Windows 8, 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.

Next, you will be asked where to install the server (figure 2.2). 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.

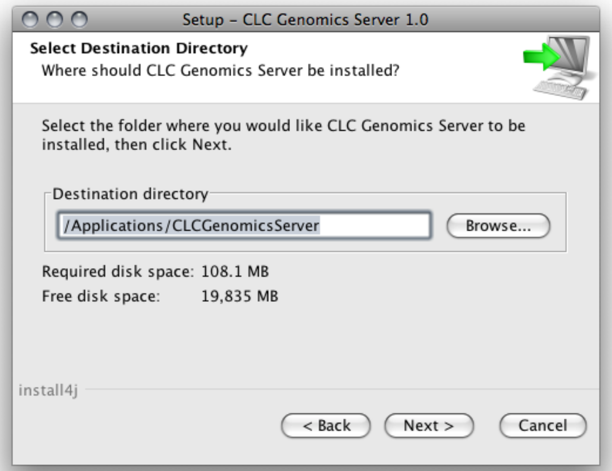
The installer allows you to specify the maximum amount of memory the CLC Server will be able to utilize (figure 2.3). The range of choice depends on the amount of memory installed on your


Enter Password

Please enter your password:

You must be a member of the Administrator group

Figure 2.1: *Enter your password.*


Setup – CLC Genomics Server 1.0

Select Destination Directory

Where should CLC Genomics Server be installed?

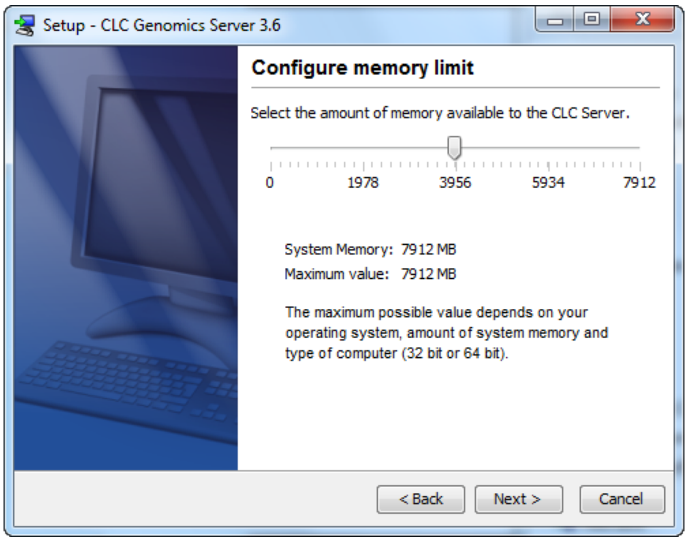
Select the folder where you would like CLC Genomics Server to be installed, then click Next.

Destination directory: /Applications/CLCGenomicsServer

Required disk space: 108.1 MB

Free disk space: 19,835 MB

Figure 2.2: *Choose where to install the server. Exemplified here with* **CLC Genomics Server**


Setup - CLC Genomics Server 3.6

Configure memory limit

Select the amount of memory available to the CLC Server.

System Memory: 7912 MB

Maximum value: 7912 MB

The maximum possible value depends on your operating system, amount of system memory and type of computer (32 bit or 64 bit).

Figure 2.3: *Choose the maximum amount of memory used by the server.*

system and on the type of machine used. If you do not have a reason to change this value you should simply leave it at the default setting.

If you are installing the Server on a Linux or Mac system, you are offered the option to specify a user account that will be used to run the *CLC Server* process. Having a specific, non-root user for this purpose is generally recommended. On a standard setup, this would have the effect of adding this username to the service scripts, which can then be used for starting up and shutting down the *CLC Server* service and setting the ownership of the files in the installation area. Downstream, the user running the *CLC Server* process will own files created in File Locations, for example, after data import or data analyses.

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.On Linux or Mac, if you have chosen the option to start the system at the end of installation, the service should also have started running. Please note that if you do not already have a license file installed, then the *CLC 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 Server* service is provided in section 2.8.

2.3 Installation modes - console and silent

Console mode and silent mode are available when launching Workbench installers on the command line.

Console mode can be particularly useful when installing on remote systems. On Linux, this mode is enabled by using the option `-c`. On Windows the option is `-console`.

Silent mode allows the installation to be run in a hands off manner. Default answers to all prompts will be used. Silent mode is activated using the `-q` parameter. On Windows, the `-console` option can be appended *as the second parameter*, to ensure output to the console.

Specify an installation directory in combination with silent mode to specify a non-default installation directory and still not be prompted for responses to questions. The installation directory is specified using the `-dir` option.

So, on a Windows system, running the installation in silent mode, with console output and specifying the directory to install to as "c:\bioinformatics\clc" would look like:

```
CLCGenomicsServer_7_5.exe -q -console -dir "c:\bioinformatics\clc"
```

On a Linux system, a similar command to install to a directory "/opt/clcgenomicsworkbench9" would look like:

```
CLCGenomicsServer_7_5.exe -c -q -dir /opt/clcgenomicsworkbench9
```

Note for Windows: 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.

2.4 Biomedical Genomics Server Extension - installation notes

The *Biomedical Genomics Server Extension* is based on *CLC Genomics Server*. To extend the functionality of the *CLC Genomics Server* with Biomedical Genomics specific tools you will need to download an additional license for the *CLC Genomics Server* - the 'Biomedical Genomics Server Extension' license. To download the license, please follow the instructions for download of the regular Genomics Server license you used to first download the Genomics Server license (see section 2.7). When this has been done, you can proceed directly, without restarting the server, to the installation of the 'Biomedical Genomics Server Extension' license by following the same procedure.

When you have an installed *CLC Genomics Server* with a Biomedical Genomics Server Extension there are a few more things you must do before you are ready to start using the server:

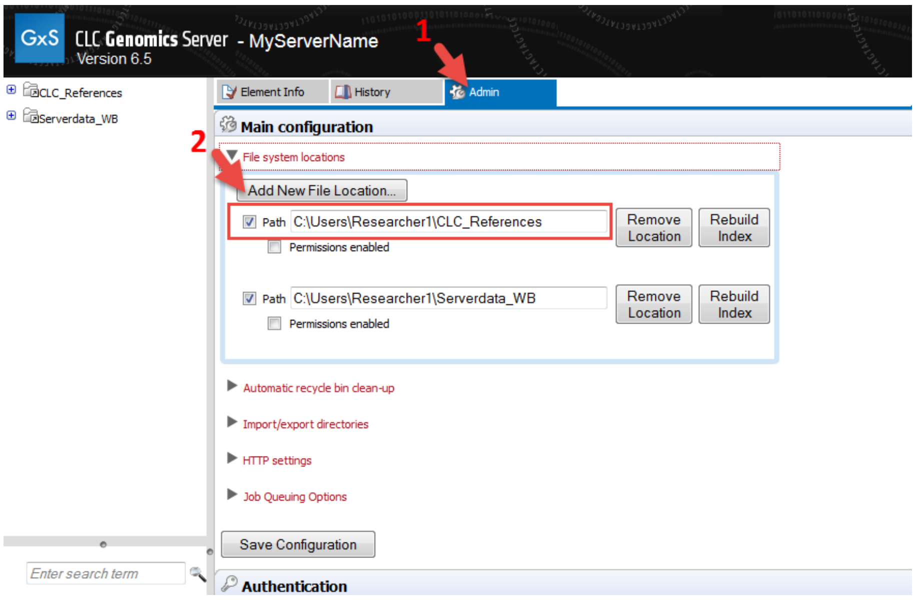
2.4.1 On the server: Create reference data directory

- You must create a directory on the server hard drive. This directory is where the Biomedical Genomics Workbench reference data will be stored. Please note that the name of the directory must be "CLC_References".

2.4.2 Using the *CLC Genomics Server* web interface: Add reference data location

- Open the CLC Genomics Server web interface and add a new file location, which is the path to the directory you have just created.

  To do this click on the "Admin" tab, then "Main configuration", select "File system locations", and click on the button labeled "Add New File Location" as shown in figure 2.4.



Path C:\Users\Researcher1\CLC_References

Path C:\Users\Researcher1\Serverdata_WB

Figure 2.4: *After you have created a directory you must mount the directory by going to the CLC Genomics Server web interface. This is done under the "Admin tab", "Main configuration", "File system locations", and "Add New File Location". Please note that the name of the directory must be "CLC_References" as shown in this example.*

2.5 Upgrading an existing installation

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

- Make sure that nobody is using the server (see section 7.1). 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. These maintained settings include the Data Locations, Import/Export directories, BLAST locations, Users and Groups, and External Application settings.

If you have a CLC Job Node setup, you will also need to upgrade the *CLC Server* software on each job node. Upgrading the software itself on each node is all you need to do. Configurations and plugins 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 plugin.

Second, check that all plugins installed on the *CLC Server* are up to date (see section 10). You can download updated versions of plugins from `http://www.qiagenbioinformatics.com/plugins/`.

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 *CLC Genomics Server* `http://www.qiagenbioinformatics.com/products/clc-genomics-server/latest-improvements/current-line/`.

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 6.2.3.

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 3.5 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.

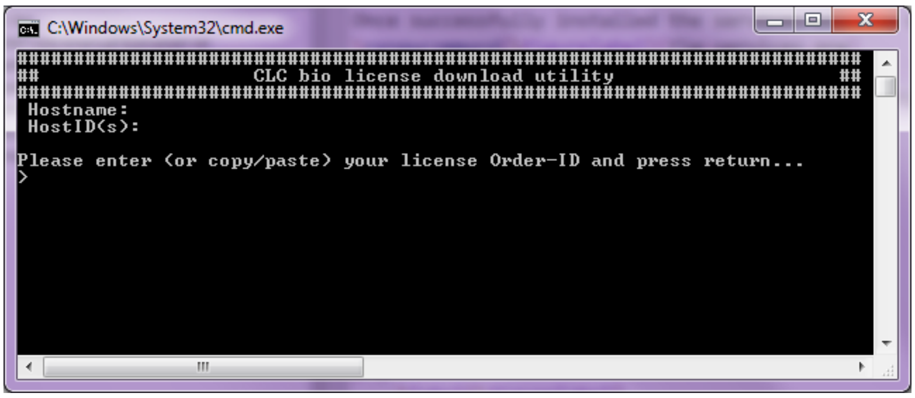
Besides the public port described above the server also uses an internal port on 7776. There is no need to allow incoming connections from client machines to this port.

2.7 Downloading a license

The *CLC Server* looks for licenses in the `licenses` folder in the installation area. Downloading and installing licenses is similar for all supported platforms, but varies in certain details. Please check the platform-specific instructions below for how to download a license file on the system you are running the *CLC Server* on or the section on downloading a license to a non-networked machine if the *CLC Server* is running on a machine without a direct connection to the external network.

2.7.1 Windows license download

License files are downloaded using the `licensedownload.bat` 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.



```
################################################################################
##                      CLC bio license download utility                      ##
################################################################################
 Hostname:
 HostID(s):

Please enter (or copy/paste) your license Order-ID and press return...
>
```

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

Paste the Order ID supplied by QIAGEN (right-click to **Paste**) and press Enter. Please contact AdvancedGenomicsSupport@qiagen.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).

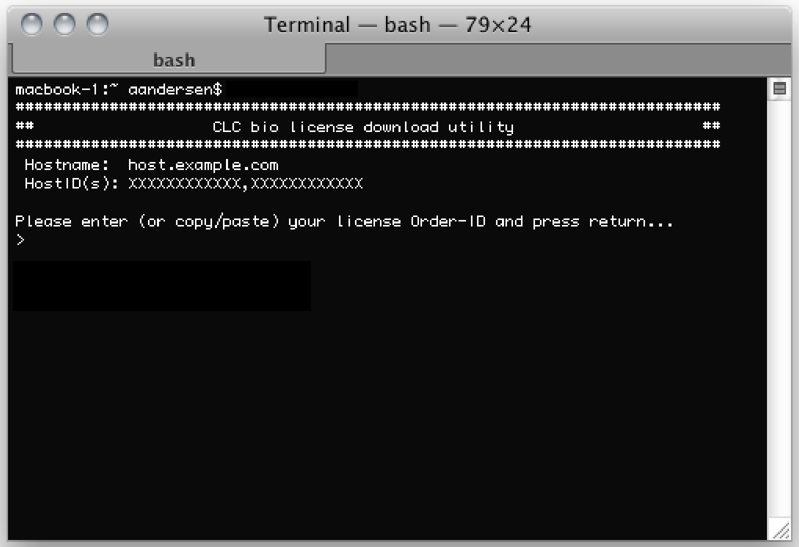
2.7.2 Mac OS license download

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 QIAGEN and press Enter. Please contact AdvancedGenomicsSupport@qiagen.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).



```
macbook-1:~ aandersen$ [redacted]
###############################################################################
##                       CLC bio license download utility                    ##
###############################################################################
 Hostname:  host.example.com
 HostID(s): XXXXXXXXXXXX,XXXXXXXXXXXX

Please enter (or copy/paste) your license Order-ID and press return...
>
[redacted]
```

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 QIAGEN Aarhus. Please contact AdvancedGenomicsSupport@qiagen.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 (see how to restart the server in section 2.8.3).

2.7.4 Download a static license on a non-networked machine

To download a static license for a machine that does not have direct access to the external network, you can follow the steps below after the Server software has been installed.

- Determine the host ID of the machine the server will be running on by running the same tool that would allow you to download a static license on a networked machine. The name of this tool depends on the system you are working on:
  - Linux: downloadlicense
  - Mac: downloadlicense.command
  - Windows: licensedownload.bat

  When you run the license download tool, the host ID for the machine you are working on will be printed to the terminal.

  In the case of a job node setup, the only machine you need the host ID for is the master node. This is the machine the license file will be stored on.

- Make a copy of this host ID such that you can use it on a machine that has internet access.
- Go to a computer with internet access, open a browser window and go to the relevant network license download web page:

  For the Genomics Server version 5.0 or higher, please go to:

  `https://secure.clcbio.com/LmxWSv3/GetServerLicenseFile`

  For the Biomedical Genomics Server add-on (all versions) please go to:

  `https://secure.clcbio.com/LmxWSv3/GetLicenseFile`

  For the Genomics Server version 4.5.2 and lower, please go to:

  `http://licensing.clcbio.com/LmxWSv2/GetServerLicenseFile`

  It is vital that you choose the license download page appropriate to the version of the software you plan to run.
- Paste in your license order ID and the host ID that you noted down earlier into the relevant boxes on the webpage.
- Click on 'download license' and save the resulting .lic file.
- Take this file to the machine acting as the CLC Server master node and place it in the folder called 'licenses' in the CLC Server installation directory.
- Restart the CLC Server software.

2.8 Starting and stopping the server

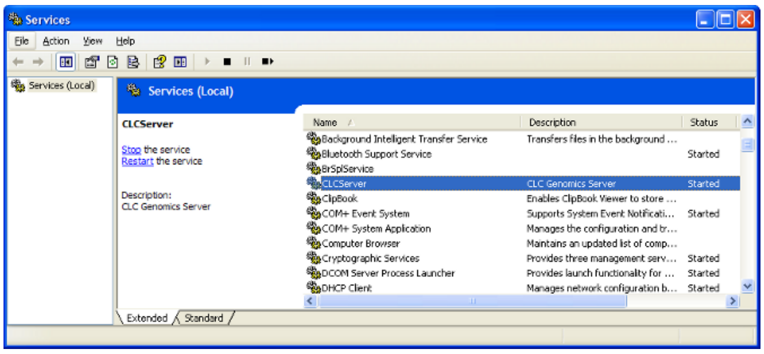
2.8.1 Microsoft Windows

On Windows based systems the *CLC Server* can be controlled through the *Services* control panel.

Depending on your server solution, the service is named:

- *CLC Genomics Server*: CLCGenomicsServer
- *Biomedical Genomics Server Extension*: CLCGenomicsServer

Choose the service and click the start, stop or restart link as shown in figure 2.7.



CLCServer

Stop the service
Restart the service

Description:
CLC Genomics Server

Name	Description	Status
Background Intelligent Transfer Service	Transfers files in the background ...	
Bluetooth Support Service		Started
BrSplService		
CLCServer	CLC Genomics Server	Started
ClipBook	Enables ClipBook Viewer to store ...	
COM+ Event System	Supports System Event Notificati...	Started
COM+ System Application	Manages the configuration and tr...	
Computer Browser	Maintains an updated list of comp...	
Cryptographic Services	Provides three management serv...	Started
DCOM Server Process Launcher	Provides launch functionality for ...	Started
DHCP Client	Manages network configuration b...	Started

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

Once your server is started, you can use the Admin tab on the server web interface to manage your server operation (see section 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.

Remember to replace *CLCServer*, in the commands listed below, with the name from the following list, corresponding to your server solution:

- *CLC Genomics Server*: CLCGenomicsServer
- *Biomedical Genomics Server Extension*: CLCGenomicsServer

To start the server run the command:

```
sudo ./CLCServer start
```

To stop the server run the command:

```
sudo ./CLCServer stop
```

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

```
sudo ./CLCServer 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.

Once your server is started, you can use the Admin tab on the server web interface to manage your server operation (see section 7).

2.8.3 Linux

You can start and stop the *CLC Server* service from the command line. You can also configure the service to start up automatically after the server machine is rebooted.

During installation of the *CLC Server* a service script is placed in /etc/init.d/.

This script will have a name reflecting the server solution, and it includes the name of the custom user account specified during installation for running the *CLC Server* process.

Starting and stopping the service using the command line:

To start the *CLC Server*:

```
sudo service CLCGenomicsServer start
```

To stop the *CLC Server*:

```
sudo service CLCGenomicsServer stop
```

To restart the *CLC Server*:

```
sudo service CLCGenomicsServer restart
```

To view the status of the *CLC Server*:

```
sudo service CLCGenomicsServer status
```

Start service on boot up:

On Red Hat Enteprise Linux and SuSE this can be done using the command:

```
sudo chkconfig CLCGenomicsServer on
```

How to configure a service to automatically start on reboot depends on the specific Linux distribution. Please refer to your system documentation for further details.

Troubleshooting

If the *CLC Server* is run as a service as suggested above, then the files in the installation area of the software and the data files created after installation in CLC Server File Locations will be owned by the user specified to run the *CLC Server* process. If someone starts up the *CLC Server* process as root (i.e. an account with super-user privileges) then the following steps are recommended to rectify the situation:

1. Stop the *CLC Server* process using the script located within the installation area of the *CLC Server* software. You can do that using the full path to this script, or by navigating to the installation area and running:

   ```
   sudo ./CLCGenomicsServer stop
   ```

2. Change ownership recursively on all files in the installation area of the software and on all areas specified as Server File Locations.

3. Start the *CLC Server* service as the specified user by using the service script:

   ```
   sudo service CLCGenomicsServer start
   ```

4. In case the server still fails to start correctly it can be started in the foreground with output being written to the console to help identify the problem. It is done by running:

   ```
   sudo ./CLCGenomicsServer start-launchd
   ```

Once your server is started, you can use the Admin tab of the web administrative interface to manage your server operation (see section 7).

2.9 Installing relevant plugins in the Workbench

To access the *CLC Server* from a CLC Workbench, the CLC Workbench Client Plugin must be installed in the Workbench. This will allow you to log into the CLC Server, access data from CLC Server file and data locations and submit analyses to your CLC Server.

Plugins are installed in a Workbench using the Plugins Manager[1], which can be accessed via the menu in the Workbench

Help | Plugins... ( )

[1] In order to install plugins 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".

or via the **Plugins ( )** button on the Toolbar.

From within the Plugins manager, choose the Download Plugins tab and click on the relevant plugin or module. Then click on the button labeled **Download and Install**.

If you are working on a system not connected to the internet, then you can also install the plugin or module by downloading the cpa file from the plugins page of our website:

http://www.qiagenbioinformatics.com/plugins/

Then start up the Plugin manager within the Workbench, and click on the button at the bottom of the Plugin manager labeled **Install from File**.

You need to restart the Workbench before the plugin is ready for use.

If you want users to be able to use **External applications** (see chapter 12) on the server, the CLC External Applications Plugin needs to be installed in the Workbench the same way as described above.

2.10 Installing the database

For *CLC Server* solutions where the license includes the add-on *CLC Bioinformatics Database*, support for data management in an SQL-type database is available. This section describes how to install and setup *CLC Bioinformatics Database* for the *CLC Server*.

2.10.1 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. *CLC Bioinformatics Database* can be used with a number of different DMBS implementations. 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/

In the case of MySQL and Oracle, you will need to have the appropriate JDBC driver and this will need to be placed in the `userlib` folder of the CLC software installation area. See section 15.3 for further details on this as well as additional guidance for special configurations for DBMSs.

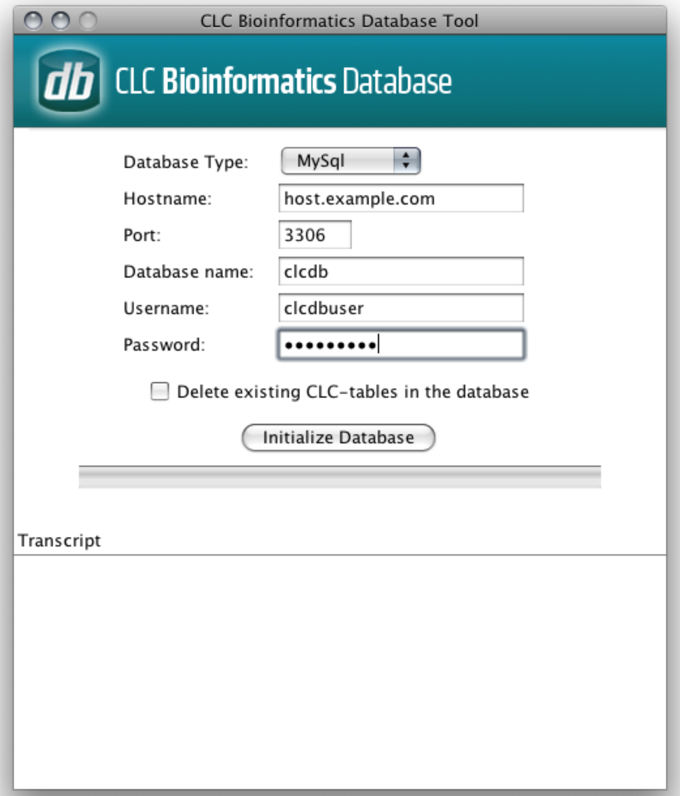
2.10.2 Create a new database and user/role

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.

2.10.3 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 (see figure 2.8).



Database Type: MySql
Hostname: host.example.com
Port: 3306
Database name: clcdb
Username: clcdbuser
Password: ••••••••••
☐ Delete existing CLC-tables in the database

Figure 2.8: *The CLC Bioinformatics Database tool*

- Download the CLC Bioinformatics Database Tool from `http://www.qiagenbioinformatics.com/product-downloads/`
- 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 more information. If you need assistance, please contact AdvancedGenomicsSupport@qiagen.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.

Chapter 3

Basic configuration

3.1 Logging into the administrative interface

The administrative interface for a running *CLC Server* is accessed via a web browser. Most configuration occurs via this interface. Simply type the host name of the server machine you have installed the *CLC 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/` or `http://localhost:7777/`

The default administive user credentials are:

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

Use these details the first time you log in. We recommend that you change this password.

Details of how to change the administrative user password is covered in section 4.1.

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 are simple pointers to folders on the file system (section 3.2.1).

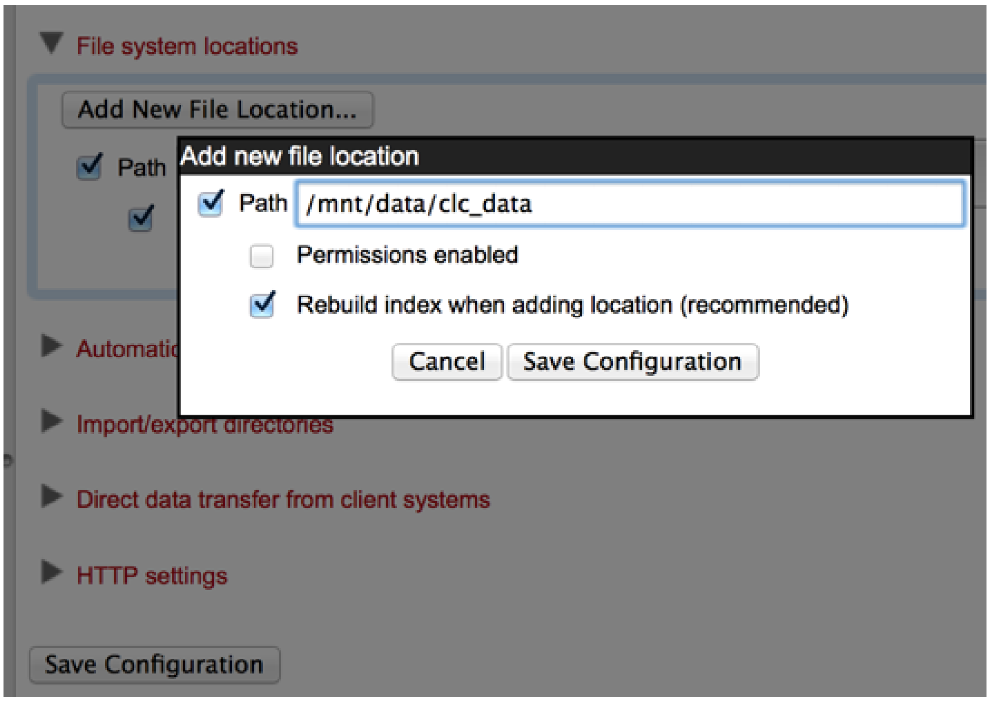
For *CLC Server* solutions where the license includes the add-on *CLC Bioinformatics Database*, the location can alternatively be based on a *CLC Bioinformatics Database* (section 3.2.2).

3.2.1 Adding a file system location

To set up a file system 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.

Under the **File system locations** heading, click the **Add New File Location** button to add a new file system location (see figure 3.1).



Add new file location

✓ Path /mnt/data/clc_data

☐ Permissions enabled

✓ Rebuild index when adding location (recommended)

Cancel | Save Configuration

Figure 3.1: *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 left hand side of the window in the server **Navigation Area**. 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 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 Server*. Such areas should be directly accessed only by the *CLC 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 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 more about job node setups in section 6.

3.2.2 Adding a database location

To add a database location to the server, the server license should include the add-on *CLC Bioinformatics Database*.

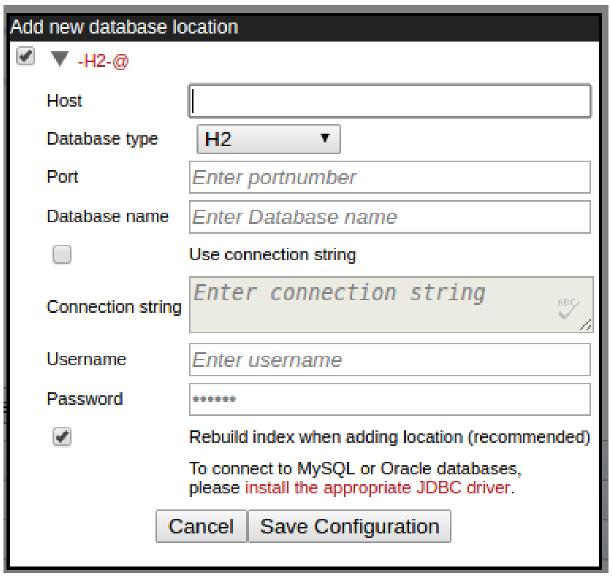
Before adding a database location, you need to set up the database itself. This is described in section 2.10.

To set up a database location on the CLC Server,

- Open a web browser and navigate to the web administrative interface.
- Go to the *Admin* tab and open the *Main configuration* section.
- Under the **Database locations** heading, click the **Add New Database Location** button (figure 3.2). Enter the required information about host, port and type of database. A connection string is generated from this. A custom connection string can be entered instead. The user name and password refers to the user role on your Database Management System (DBMS), see section 2.10.

  There are two versions of Oracle in the drop down list of database types. One is the traditional one, which uses the SID style (e.g. `jdbc:oracle:thin:@[HOST][:PORT]:SID`). The other uses the thin-style service name (e.g. `jdbc:oracle:thin:@//[HOST][:PORT]/SERVICE`).
- Click the *Save Configuration* button to save the configuration.

The added database location should now appear in the **Navigation Area** in the left hand side of the window.


Add new database location

✓ -H2-@

Host	
Database type	H2
Port	Enter portnumber
Database name	Enter Database name
☐	Use connection string
Connection string	Enter connection string
Username	Enter username
Password	••••••
✓	Rebuild index when adding location (recommended)

To connect to MySQL or Oracle databases, please install the appropriate JDBC driver.

Figure 3.2: *Add new database location.*

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 found in the **Status and Management** tab page 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 situations when it is beneficial to be able to interact with (non-CLC) files directly on your server filesystem.

A common use case would be importing high-throughput sequencing data or large molecule libraries from folders where it is stored on the same system that your *CLC Server* is running on. This could eliminate the need for each user to copy large data files to the machine the CLC Workbench is running on before importing the data into a *CLC Server* data area.

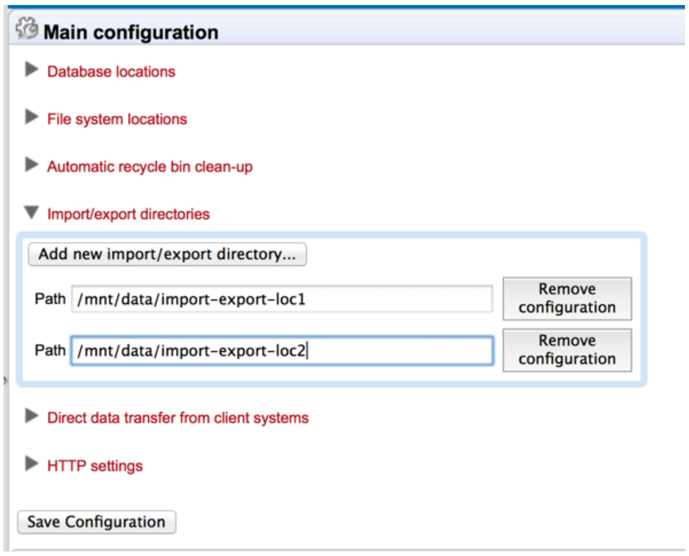
Another example is if you wish to export data from CLC format to other formats and save those files on your server machine's filesystem (as opposed to saving the files in the system your Workbench is running on).

From the administrator's point of view, this is about configuring folders that are safe for the *CLC*

Server to read and write to on the server machine system.

This means that users logged into the *CLC Server* from their Workbench will be able to access files in that area, and potentially write files to that area. Note that the *CLC 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.


Main configuration
Database locations
File system locations
Automatic recycle bin clean-up
Import/export directories
Add new import/export directory...
Remove configuration
Remove configuration
Direct data transfer from client systems
HTTP settings
Save Configuration

Path /mnt/data/import-export-loc1

Path /mnt/data/import-export-loc2

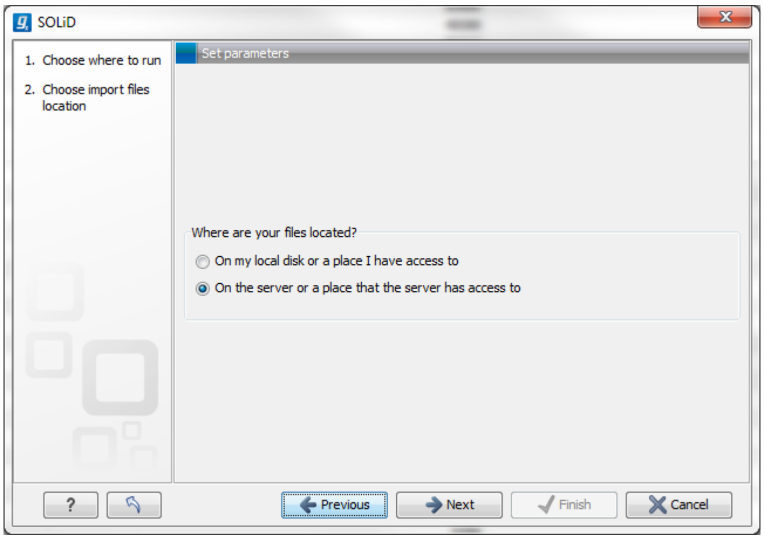
Figure 3.3: *Folders on the server that should be available from the Workbench for browsing, importing data from and exporting data to.*

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. If a user, that is logged into the *CLC Server* via their CLC Workbench, wishes to import e.g. high throughput sequencing data, an option like the one 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 administator has configured as an Import/export location (an example is shown in figure 3.5).

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 *CLC Server*, or BLAST databases.


SOLiD

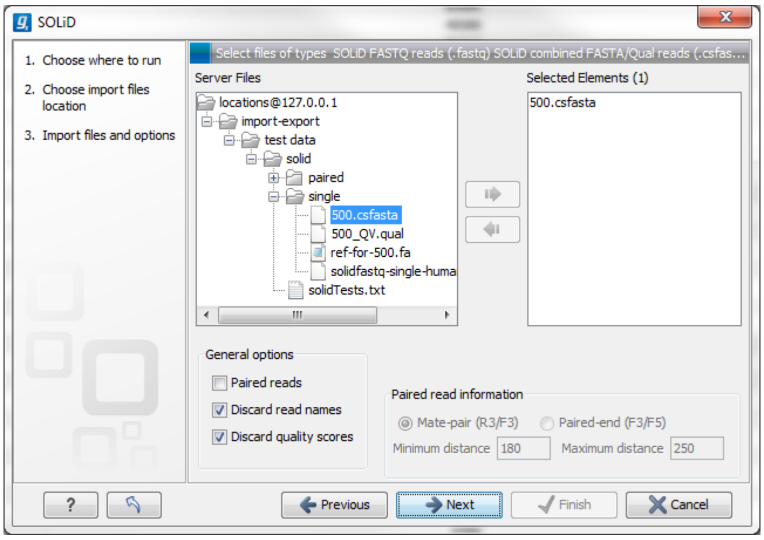
1. Choose where to run
2. Choose import files location

Set parameters

Where are your files located?

- On my local disk or a place I have access to
- [✓ On the server or a place that the server has access to]

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


SOLiD

1. Choose where to run
2. Choose import files location
3. Import files and options

Select files of types SOLiD FASTQ reads (.fastq) SOLiD combined FASTA/Qual reads (.csfas...

Server Files

```
locations@127.0.0.1
  import-export
    test data
      solid
        paired
        single
          500.csfasta
          500_QV.qual
          ref-for-500.fa
          solidfastq-single-huma
        solidTests.txt
```

Selected Elements (1)

500.csfasta

General options

- Paired reads
- [✓ Discard read names]
- [✓ Discard quality scores]

Paired read information

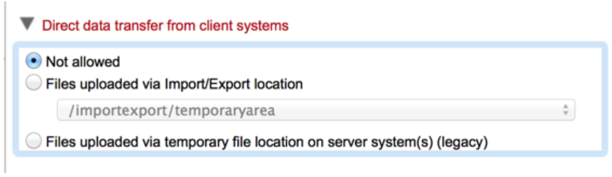
- [✓ Mate-pair (R3/F3)]
- Paired-end (F3/F5)

Minimum distance 180 Maximum distance 250

Figure 3.5: *Selecting files on server file system.*

3.4 Direct data transfer from client systems

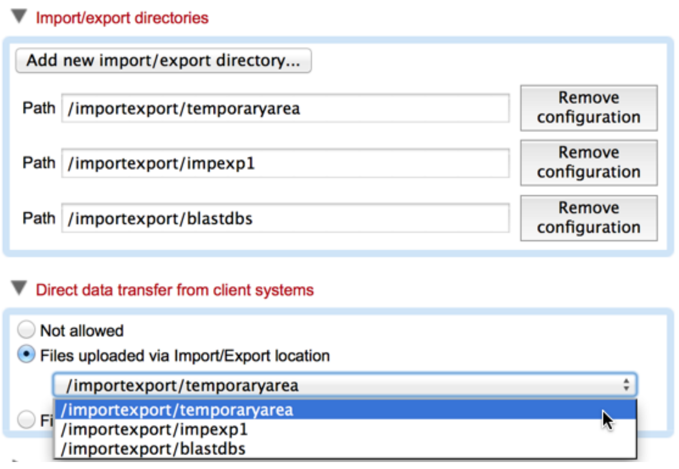
Users of client systems are able, by default, to import data from a client system that the CLC Workbench or CLC Command Line Tools is installed on directly into a Server file or data location. The settings shown in figure 3.6 control whether this facility should be allowed and, if it should, then how the temporary data associated with Server data import should be handled.


Direct data transfer from client systems

- [✓ Not allowed]
- Files uploaded via Import/Export location
  /importexport/temporaryarea
- Files uploaded via temporary file location on server system(s) (legacy)

Figure 3.6: *Specify whether data from client systems can be directly imported into Server locations and if so, where temporary data associated with these actions should be located.*

If this facility is allowed on a system, we recommend the option to use a specified Import/Export location. To use this option, an Import/Export area must first be defined. It will then be available from the drop down list of areas to choose from (figure 3.7).


Import/export directories
Add new import/export directory...
Remove configuration
Direct data transfer from client systems

Path /importexport/temporaryarea
Path /importexport/impexp1
Path /importexport/blastdbs

Not allowed
[✓ Files uploaded via Import/Export location]
/importexport/temporaryarea
/importexport/temporaryarea
/importexport/impexp1
/importexport/blastdbs

Figure 3.7: *Specifying an Import/Export area for temporary data associated with the direct transfer of data from a client system into a CLC Server location.*

The use of default system temporary areas is deprecated and may be retired in future. We recommend that one of the other two options is chosen.

3.5 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).
- Once the service is restarted, please log into the administrative interface and change the default port number in the "Master node port" field under Admin | Job distribution | Server setup, then click on **Save Configuration** button to save the new setting.

3.6 Changing the tmp directory

The *CLC 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 Server can be written.

In the *server installation directory* you will find a file called `CLCServer.vmoptions`, where CLCServerwill be the name of your particular CLC server:

- *CLC Genomics Server*: CLCGenomicsServer
- *Biomedical Genomics Server Extension*: CLCGenomicsServer

Open the 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.

3.6.1 Job node setup

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 `CLCServer.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.

3.7 Setting the amount of memory available for the JVM

When running the *CLC 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:

```
-Xmx8192m
```

The number (8192) is the amount of memory in megabytes the *CLC Server* is allowed to use. This file is located in the installation folder of the *CLC 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 Server* if you wish to raise or lower the amount of RAM allocated to the Java Virtual Machine.

3.8 Limiting the number of cpus available for use

A number of the algorithms in the *CLC 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 Server* installation directory.

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

maxcores = 1

Restart the *CLC 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 Server* is allowed to use. Please note that this is not a guarantee that the *CLC 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`.

3.9 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).

3.10 Deployment of server information to CLC Workbenches

See the *Deployment manual* at `http://www.qiagenbioinformatics.com/support/manuals/` for information on pre-configuring the server log-in information when Workbench users log in for the first time.

Chapter 4

Managing users and groups

4.1 Logging in the first time and changing the 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).

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).

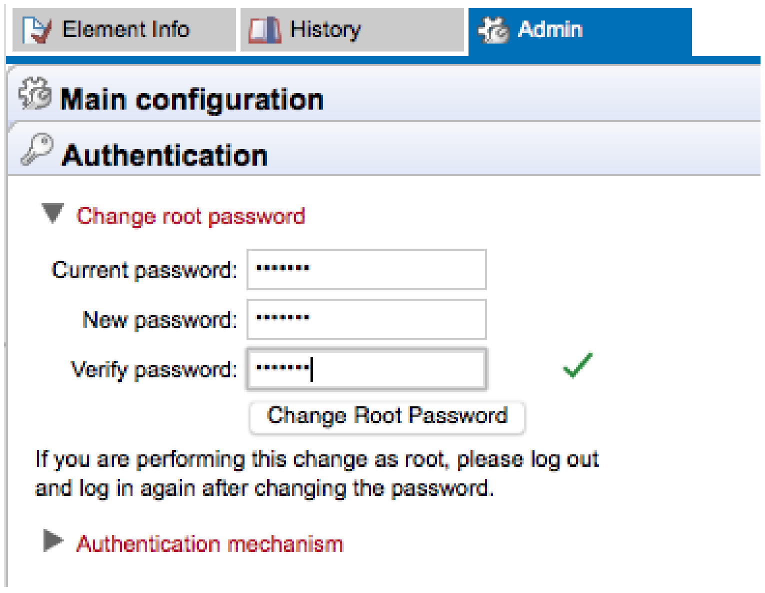
Once logged in, you can specify how the general user authentication should be done:

Admin ( ) | Authentication ( ) Authentication mechanism

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

For the LDAP and Active Directory, a settings panel will be revealed when the option is chosen, allowing you to specify the details of the integration (see figure 4.3 for an example of LDAP settings).

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



▼ Change root password

Current password: •••••••

New password: •••••••

Verify password: ••••••• ✓

If you are performing this change as root, please log out and log in again after changing the password.

▶ Authentication mechanism

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

box labeled **Admin group name** and adding any users who should be administrators of the CLC Server to this group.

4.2.1 Authentication options

Built-in authentication

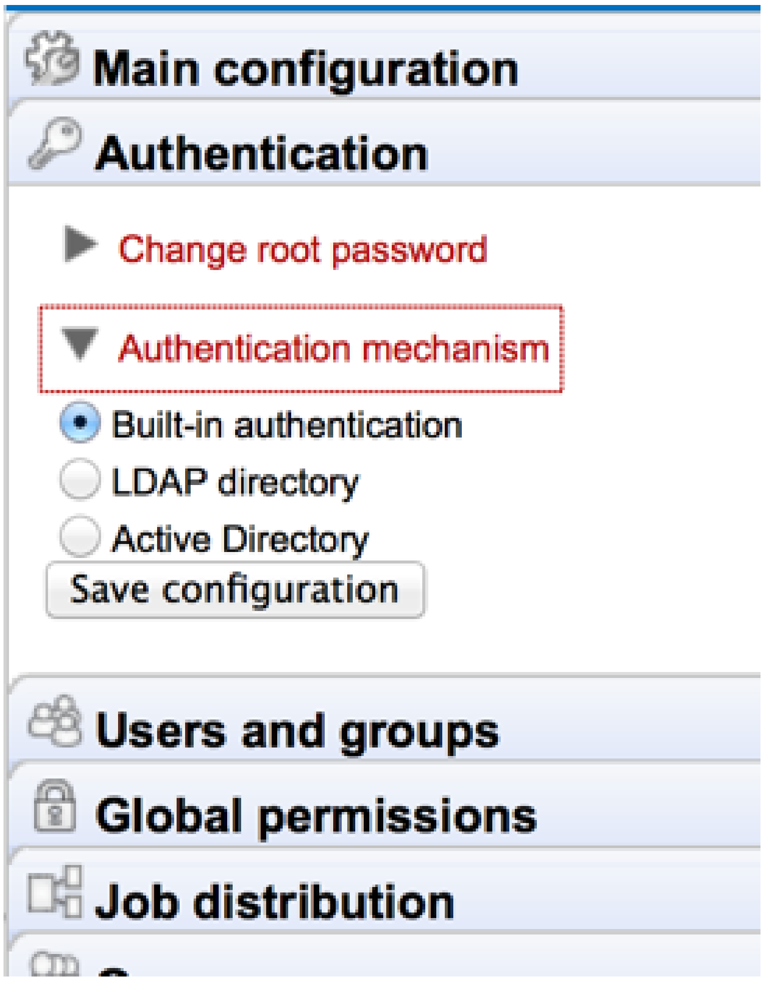
This option will enable you to set up user authentication using the server's built-in user management system. This means that you create users, set passwords, assign users to groups and manage groups using the web interface (see section 4.2.2) or using the Workbench (see section 4.3.1). All the user 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. Encryption options (Start TLS and LDAP over SSL) are available. If your LDAP server uses a certificate that is not generally trusted by the server system that the CLC Server software is running on, then it must be added to the truststore of the CLC Server installation (`CLC_SERVER_BASE/jre/lib/security/cacerts`, where `CLC_SERVER_BASE` is the server installations root location). This can be done with the keytool shipped with Java installations (also available in the `CLC_SERVER_BASE/jre/bin/keytool`), with a command like:

```
CLC_SERVER_BASE/jre/bin/keytool -import -alias \
  ldap_certificate -file LDAP_CERTIFICATE.cer -keystore \
  CLC_SERVER_BASE/jre/lib/security/cacerts -storepass changeit
```



[● Built-in authentication]
○ LDAP directory
○ Active Directory

Figure 4.2: *Three modes of user authentication.*

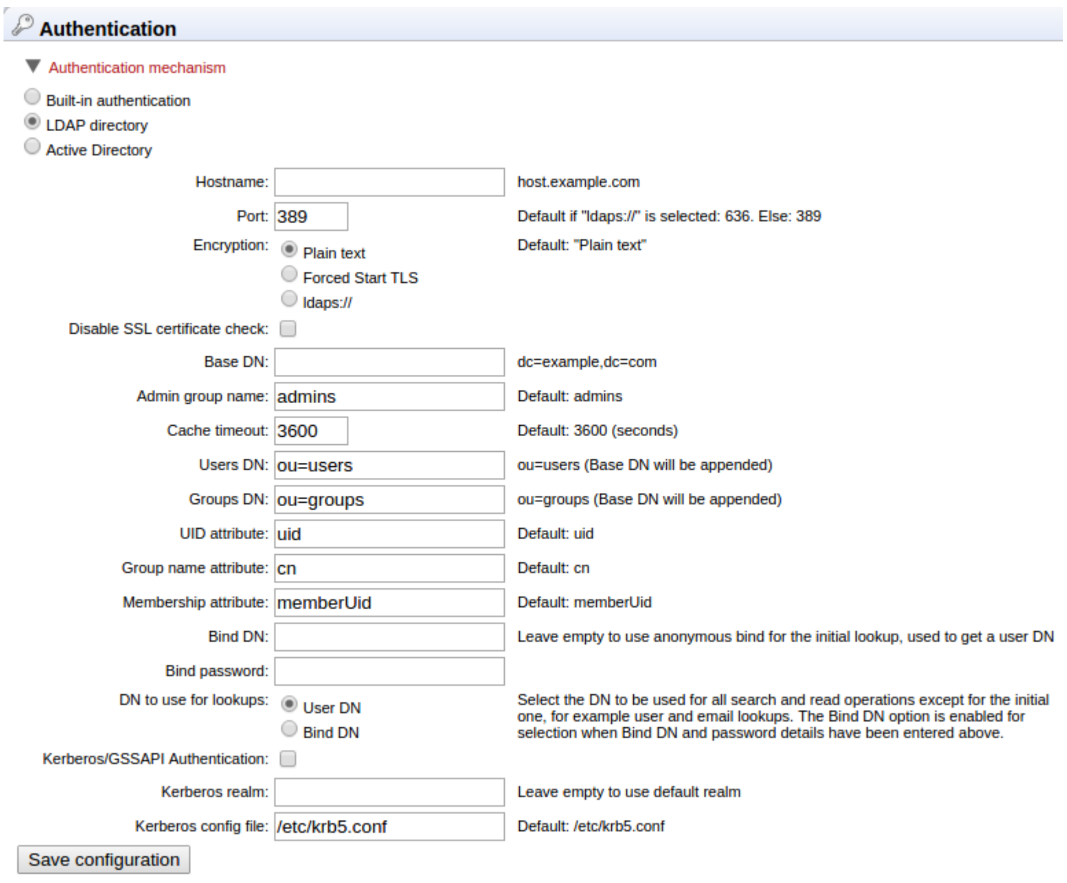
Replace `LDAP_CERTIFICATE` with the path to the certificate your LDAP server uses for Start TLS/LDAPS connections. Replace `CLC_SERVER_BASE` with the path to the servers installation location.
For a node setup, this must be done for all job nodes as well.

Caution: If you update the server installation or reinstall the server, all imported certificates will be removed, and have to be imported again. You should also be aware that certificates have an expiration date, and will not be trusted after this date. Make sure to add a new certificate in advance of the expiration date.

The **DN to use for lookups** configuration allows you to choose which bind should be used for read and search operations. If no bind DN have been entered an unauthenticated bind will be used to do the initial lookup (lookup of users DN based on the username), and all other read and search operations will be performed with users binds. If the **Bind DN** and **Bind password** have been filled in, you have the choice between using the 'Bind DN' or the 'User DN' for read and search operations, the 'Bind DN' will in this case always be used for the initial lookup.

Active Directory

This option will allow you to use an existing Active directory. This means that all information needed during authentication and group memberships is retrieved from the Active directory. Encryption options (Start TLS and LDAP over SSL) are available. Please see the notes about certificates in the LDAP section (see section 4.2.1) above for details.



Authentication

▼ Authentication mechanism

- Built-in authentication
- [✓ LDAP directory]
- Active Directory

Hostname:		host.example.com
Port:	389	Default if "ldaps://" is selected: 636. Else: 389
Encryption:	[✓ Plain text] Forced Start TLS ldaps://	Default: "Plain text"
Disable SSL certificate check:	☐	
Base DN:		dc=example,dc=com
Admin group name:	admins	Default: admins
Cache timeout:	3600	Default: 3600 (seconds)
Users DN:	ou=users	ou=users (Base DN will be appended)
Groups DN:	ou=groups	ou=groups (Base DN will be appended)
UID attribute:	uid	Default: uid
Group name attribute:	cn	Default: cn
Membership attribute:	memberUid	Default: memberUid
Bind DN:		Leave empty to use anonymous bind for the initial lookup, used to get a user DN
Bind password:		
DN to use for lookups:	[✓ User DN] Bind DN	Select the DN to be used for all search and read operations except for the initial one, for example user and email lookups. The Bind DN option is enabled for selection when Bind DN and password details have been entered above.
Kerberos/GSSAPI Authentication:	☐	
Kerberos realm:		Leave empty to use default realm
Kerberos config file:	/etc/krb5.conf	Default: /etc/krb5.conf

Figure 4.3: *LDAP settings panel.*

4.2.2 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 4.4.

4.2.3 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 4.5.

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

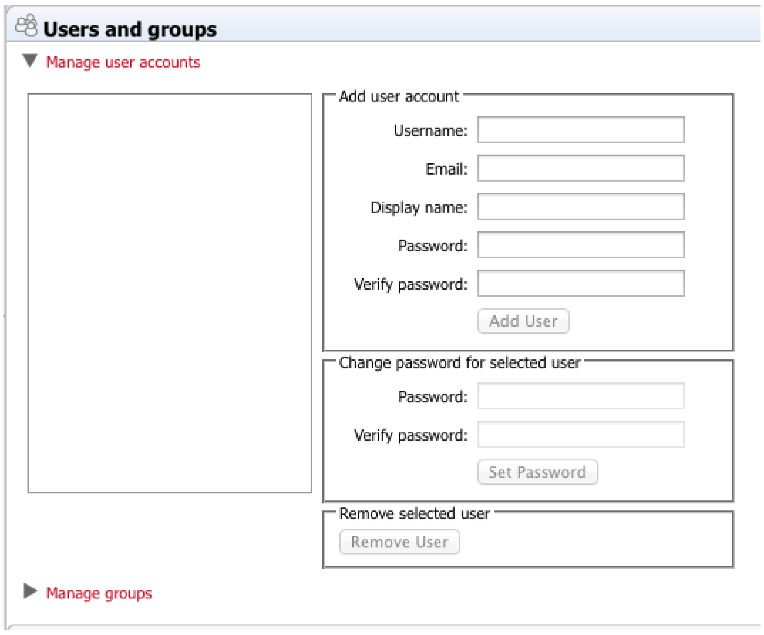


Figure 4.4: *Managing users.*

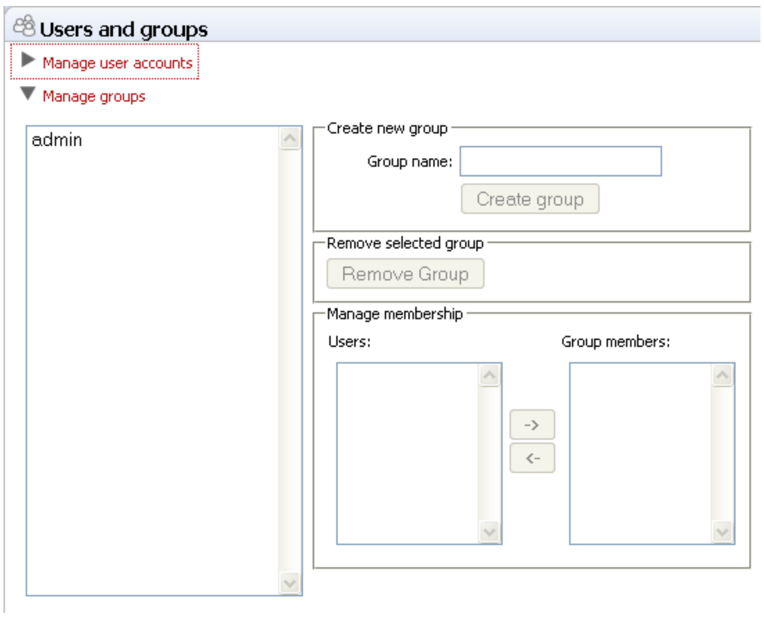


Figure 4.5: *Managing users.*

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 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 4.6.

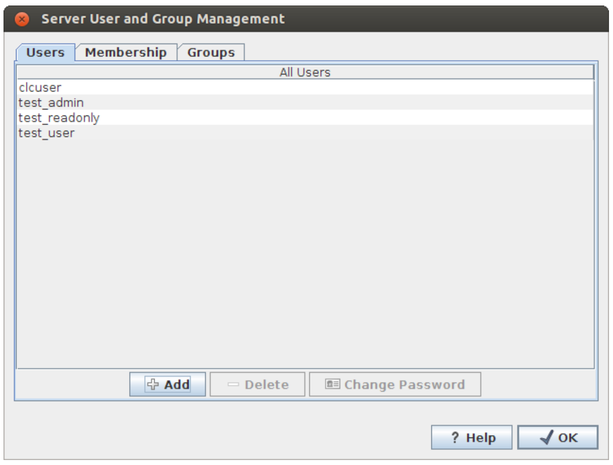


Figure 4.6: *Managing users.*

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** (▤).

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.

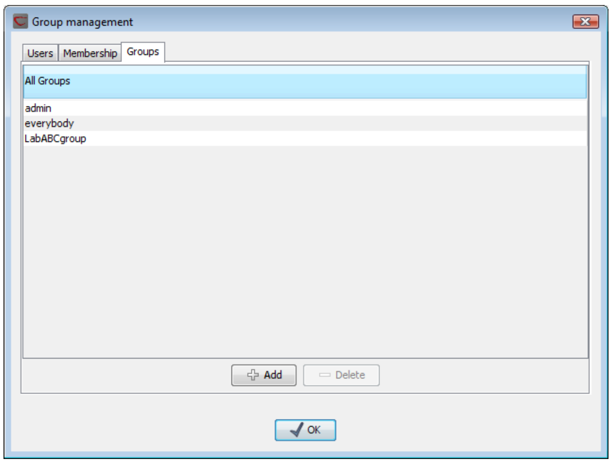
Adding and removing groups is done in the **Groups** tab (see figure 4.7).

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.

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 4.8). To the left you see a list of all users.

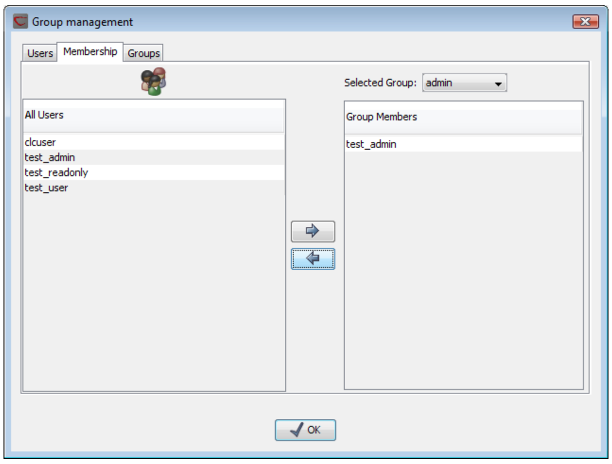
To add or remove users from a group, click the **Add** (➡) or **Remove** (⬅) buttons. To create new users, see section 4.3.1.



All Groups

admin
everybody
LabABCgroup

Figure 4.7: *Managing groups.*



Selected Group: admin

All Users

dcuser
test_admin
test_readonly
test_user

Group Members

test_admin

Figure 4.8: *Listing members of a group.*

The same user can be a member of several groups.

Chapter 5

Access privileges and permissions

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

- **Access to data in the server's file and data locations**. Common examples would be restricting access to particular folders to specified groups of users or setting reference data access to be "read-only".
- **Launching jobs** on the server can be restricted to particular groups of users. Permissions settings are available for data import, export and running particular analyses, whether built-in analyses, installed Workflows or configured External Applications. In the case of grid setups, access to particular grid presets can also be restricted to particular groups.
- **Access to the import/export directories**. Directories on the server file system configured as import/export directories can have their access via the CLC Server restricted to certain groups of users.

5.1 Controlling access to CLC Server data

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

Two types of access can be granted to a group on any folder within a server location:

Read access Users of the designated group(s) can see the elements in the folder, open them and copy from them. Access can be through any route, for example, via the CLC Command Line Tools or via the Workbench, for example when browsing in the **Navigation Area** of a Workbench, searching, or when clicking the "originates from" link in the **History** ( ) of data.

Write access Users of the designated group(s) can make and **Save** ( ) changes to an element, and new elements and subfolders can be created in that area.

For a user to be able to access a folder, they must have read access to all the folders above it in the hierarchy. In the example shown in figure 5.1, to access the *Sequences* folder, the user must have access to both the *Example Data* and *Protein* folders.

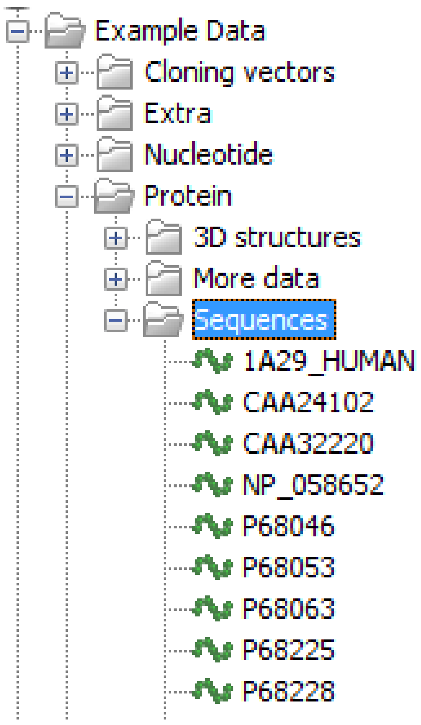


Figure 5.1: *A folder hierarchy on the server.*

It is fine to just give write access to the final folder. For example, read access only could be granted to the *Example Data* and *Protein* folders, with read and write access granted to the *Sequences* folder.

Permissions on CLC Server File Locations must be **explicitly enabled** via the web administrative interface if they are desired (see section 3.2.1). Please see 5.1.3 for further details about the system behaviour if permissions are not enabled and configured.

Configuring the permissions is done via a CLC Workbench acting as a client for the CLC Server. At the point when permissions are enabled on a File Location via the server web administrative interface, Only the *CLC Server* root user or users in a configured admin group have access to data held in that File Location at this point. No groups will have read or write access to any area under this location. Permissions should then be explicitly set by the root or other admin user on the folders in that area, as described below.

5.1.1 Setting permissions on a folder

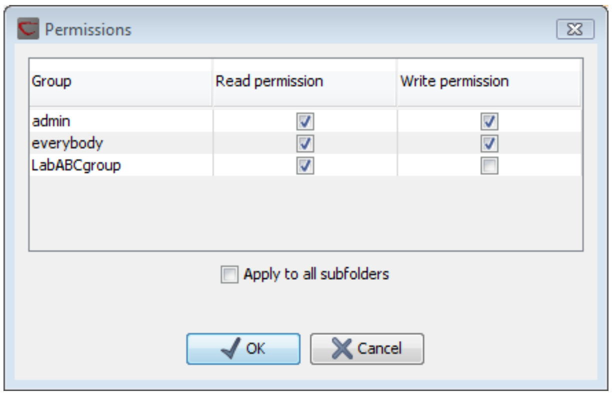
This step is done from within a CLC Workbench. Start up a copy of a CLC Workbench that has a plugin called the *CLC Server Client Plugin* installed. From within the Workbench, go to the File menu and choose the item **CLC Server Login**. Log into the CLC Server as an administrative user.

You can then set permissions on folders within File Locations that have had permissions enabled or on Database Locations, if you have a CLC Bioinformatics Database.

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**.



Group	Read permission	Write permission
admin	✓	✓
everybody	✓	✓
LabABCgroup	✓	

Figure 5.2: *Setting permissions on a folder.*

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 usually only relevant if you wish 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.

5.1.2 Recycle bin

When users delete data in the **Navigation Area** of the Workbench, it is placed in the recycle bin. When the data is situated on a data location on a *CLC Server*, the data will be placed in a recycle bin for that data location. Each user has an individual recycle bin containing the data deleted by that particular user which cannot be accessed by any other user (except server administrators, see below). This means that any permissions applied to the data prior to deletion are no longer in effect, and it is not possible to grant other users permission to see it while it is located in the recycle bin. In summary, the recycle bin is a special concept that is not included in the permission control system.

Server administrators can access the recycle bins of other users through the Workbench:

right-click the data location ( ) | Location | Show All Recycle Bins

This will list all the recycle bins at the bottom of the location as shown in figure 5.3.

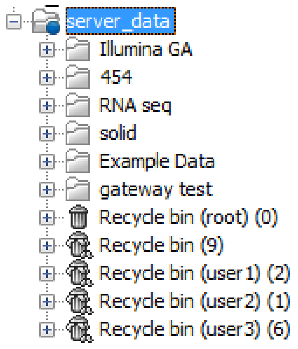


Figure 5.3: *Showing all recycle bins.*

The recycle bin without a name contains all the data that was deleted in previous versions of the *CLC Server* before the concept of a per-user recycle bin was introduced. This recycle bin can only be accessed by server administrators by selecting **Show All Recycle Bins**.

The administrator is also able to empty the recycle bin of a user:

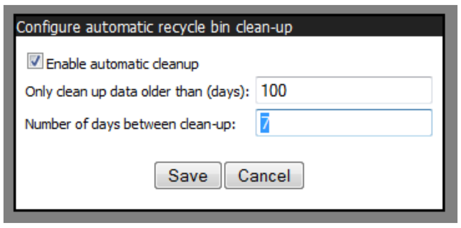
right-click the recycle bin ( ) | Empty

All recycle bins can be emptied in one go:

right-click the data location ( ) | Location | Empty All Recycle Bins

Please note that these operations cannot be undone.

CLC Server can be set to automatically empty recycle bins when the data has been there for more than 100 days. This behavior can be controlled for each data location: Under the **Main configuration** heading, click the **Automatic recycle bin clean-up** header and click the **Configure** button. This will allow you to disable the automatic clean-up completely or specify when it should be performed as shown in figure 5.4.


Configure automatic recycle bin clean-up

[✓] Enable automatic cleanup

Only clean up data older than (days): 100

Number of days between clean-up: 7

Figure 5.4: *Automatic clean-up of the recycle bin.*

Data deleted before the per-user recycle bin concept was introduced will be ignored by the automatic clean-up (this is the data located in the general recycle bin that is not labeled with a user name.

5.1.3 Technical notes about permissions and security

All data stored in *CLC Server* file system locations are owned by the user that runs the *CLC 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 Server*.

One implication of the above ownership setup is that by default, (i.e. without permissions enabled), all users logging into the *CLC 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 Server*.

Group permissions on file system locations is an additional layer within the *CLC 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 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 Server* has no effect.

5.2 Controlling access to tasks and external data

The configurations discussed in this section refer to settings under the **Global Permissions** section of the Admin tab in the CLC Server web administrative interface (figure 5.5).

Permissions can be set determining who has access to particular:

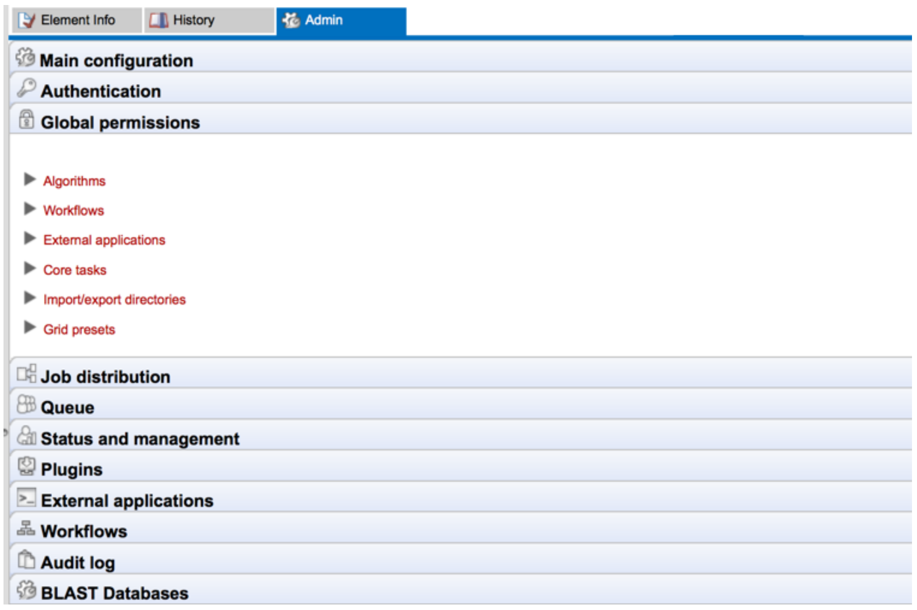


Figure 5.5: *Global permissions.*

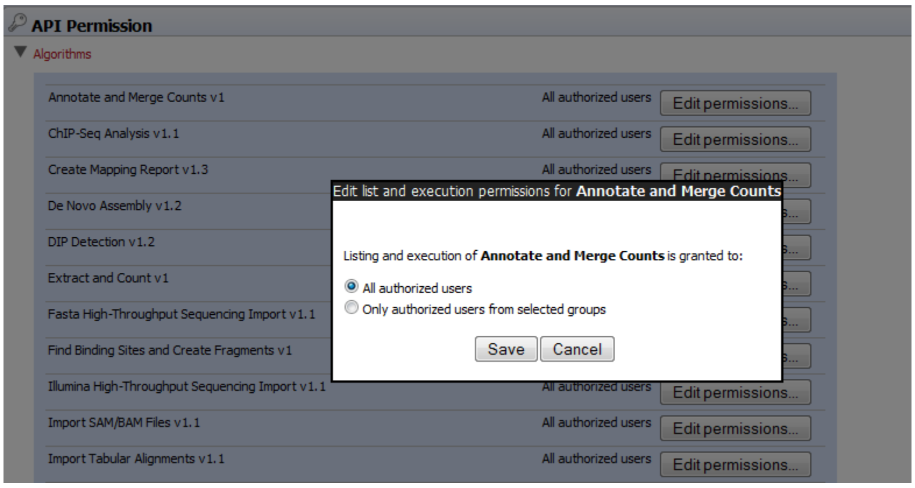
- **Algorithms** The analysis algorithms.
- **Workflows** Workflows installed on the server.
- **External** applications.
- **Core tasks** Currently covers setting permissions on actions associated with the Standard Import tools. (High throughput sequence data import is handled via tools listed in the Algorithms section.)
- **Import/export directories** File system areas not part of the CLC data setup, which the CLC Server is able to access.
- **Grid presets** For grid node setups only: presets for sending jobs to a particular queue with particular parameters. Note that grid presets are identified by name. If you change the name of a preset under the Job Distribution settings section, then this, in effect, creates a new preset. In this situation, if you had access permissions previously set, you would need to reconfigure those settings for this, now new, preset.

You can specify which groups should have access to each of the above by opening the relevant section and then clicking the **Edit Permissions** button for each relevant element listed. A dialog appears like that in figure 5.6. 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 grant or restrict access to that functionality.

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

5.3 Customized attributes on data locations

Location-specific attributes can be set on all elements stored in a given data location. Attributes could be things like company-specific information such as LIMS id, freezer position etc. Attributes are set using a CLC Workbench acting as a client to the CLC Server.



API Permission

Algorithms

Annotate and Merge Counts v1	All authorized users	Edit permissions...
ChIP-Seq Analysis v1.1	All authorized users	Edit permissions...
Create Mapping Report v1.3	All authorized users	Edit permissions...
De Novo Assembly v1.2		
DIP Detection v1.2		
Extract and Count v1		
Fasta High-Throughput Sequencing Import v1.1		
Find Binding Sites and Create Fragments v1		
Illumina High-Throughput Sequencing Import v1.1	All authorized users	Edit permissions...
Import SAM/BAM Files v1.1	All authorized users	Edit permissions...
Import Tabular Alignments v1.1	All authorized users	Edit permissions...

Edit list and execution permissions for **Annotate and Merge Counts**

Listing and execution of **Annotate and Merge Counts** is granted to:

[✓ All authorized users]
Only authorized users from selected groups

Save Cancel

Figure 5.6: *Setting permissions for an alorithm.*

Note that the attributes scheme belongs to a particular data location, so if there are multiple data locations, each will have its own set of attributes.

5.3.1 Configuring which fields should be available

To configure which fields that should be available[1] go to the Workbench:

right-click the data location | **Location** | **Attribute Manager**

This will display the dialog shown in figure 5.7.

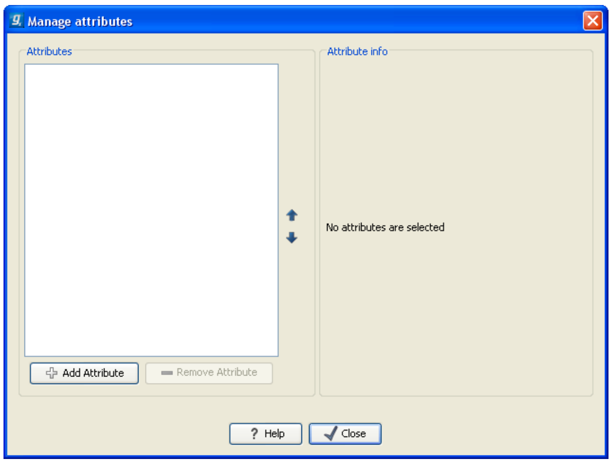


Figure 5.7: *Adding attributes.*

Click the **Add Attribute** (✚) button to create a new attribute. This will display the dialog shown in figure 5.8.

First, select what kind of attribute you wish to create. This affects the type of information that can be entered by the end users, and it also affects the way the data can be searched. The following types are available:

- **Checkbox**. This is used for attributes that are binary (e.g. true/false, checked/unchecked

[1]If the data location is a server location, you need to be a server administrator to do this

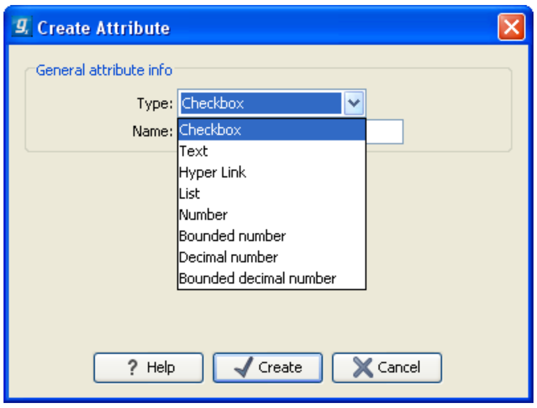


Figure 5.8: *The list of attribute types.*

and yes/no).

- **Text**. For simple text with no constraints on what can be entered.
- **Hyper Link**. This can be used if the attribute is a reference to a web page. A value of this type will appear to the end user as a hyper link that can be clicked. Note that this attribute can only contain one hyper link. If you need more, you will have to create additional attributes.
- **List**. Lets you define a list of items that can be selected (explained in further detail below).
- **Number**. Any positive or negative integer.
- **Bounded number**. Same as number, but you can define the minimum and maximum values that should be accepted. If you designate some kind of ID to your sequences, you can use the bounded number to define that it should be at least 1 and max 99999 if that is the range of your IDs.
- **Decimal number**. Same as number, but it will also accept decimal numbers.
- **Bounded decimal number**. Same as bounded number, but it will also accept decimal numbers.

When you click **OK**, the attribute will appear in the list to the left. Clicking the attribute will allow you to see information on its type in the panel to the right.

5.3.2 Editing lists

Lists are a little special, since you have to define the items in the list. When you click a list in the left side of the dialog, you can define the items of the list in the panel to the right by clicking **Add Item** (✚) (see figure 5.9).

Remove items in the list by pressing **Remove Item** (▭).

5.3.3 Removing attributes

To remove an attribute, select the attribute in the list and click **Remove Attribute** (▭). This can be done without any further implications if the attribute has just been created, but if you remove

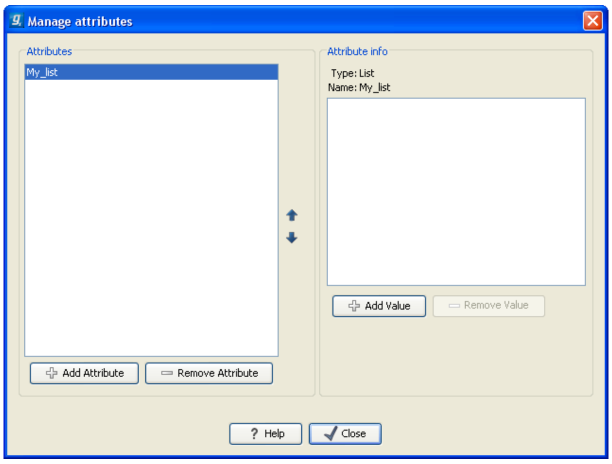


Figure 5.9: *Defining items in a list.*

an attribute where values have already been given for elements in the data location, it will have implications for these elements: The values will not be removed, but they will become static, which means that they cannot be edited anymore.

If you accidentally removed an attribute and wish to restore it, this can be done by creating a new attribute of exactly the same name and type as the one you removed. All the "static" values will now become editable again.

When you remove an attribute, it will no longer be possible to search for it, even if there is "static" information on elements in the data location.

Renaming and changing the type of an attribute is not possible - you will have to create a new one.

5.3.4 Changing the order of the attributes

You can change the order of the attributes by selecting an attribute and click the **Up** and **Down** arrows in the dialog. This will affect the way the attributes are presented for the user.

5.3.5 Filling in values

When a set of attributes has been created (as shown in figure 5.10), the end users can start filling in information.

This is done in the element info view:

> **right-click a sequence or another element in the Navigation Area | Show ( ) | Element info ( )**

This will open a view similar to the one shown in figure 5.11.

You can now enter the appropriate information and **Save**. When you have saved the information, you will be able to search for it (see below).

Note that the element (e.g. sequence) needs to be saved in the data location before you can edit the attribute values.

When nobody has entered information, the attribute will have a "Not set" written in red next to

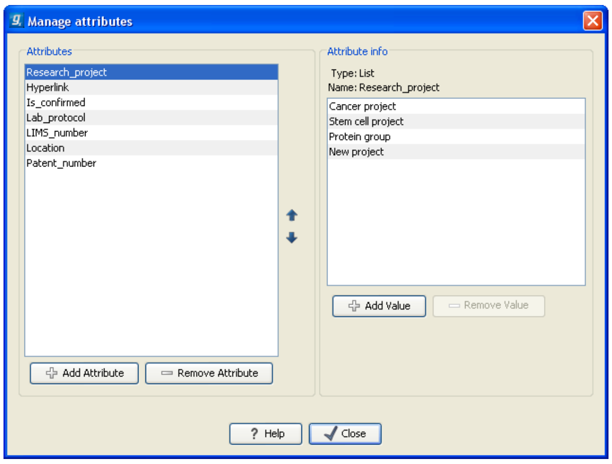


Figure 5.10: *A set of attributes defined in the attribute manager.*

the attribute (see figure 5.12).

This is particularly useful for attribute types like checkboxes and lists where you cannot tell, from the displayed value, if it has been set or not. Note that when an attribute has not been set, you cannot search for it, even if it looks like it has a value. In figure 5.12, you will *not* be able to find this sequence if you search for research projects with the value "Cancer project", because it has not been set. To set it, simply click in the list and you will see the red "Not set" disappear.

If you wish to reset the information that has been entered for an attribute, press "Clear" (written in blue next to the attribute). This will return it to the "Not set" state.

The **Folder editor**, invoked by pressing **Show** on a given folder from the context menu, provides a quick way of changing the attributes of many elements in one go (see the Workbench manuals at https://www.qiagenbioinformatics.com/support/manuals/).

5.3.6 What happens when a clc object is copied to another data location?

The user supplied information, which has been entered in the **Element info**, is attached to the attributes that have been defined in this particular data location. If you copy the sequence to another data location or to a data location containing another attribute set, the information will become fixed, meaning that it is no longer editable and cannot be searched for. Note that attributes that were "Not set" will disappear when you copy data to another location.

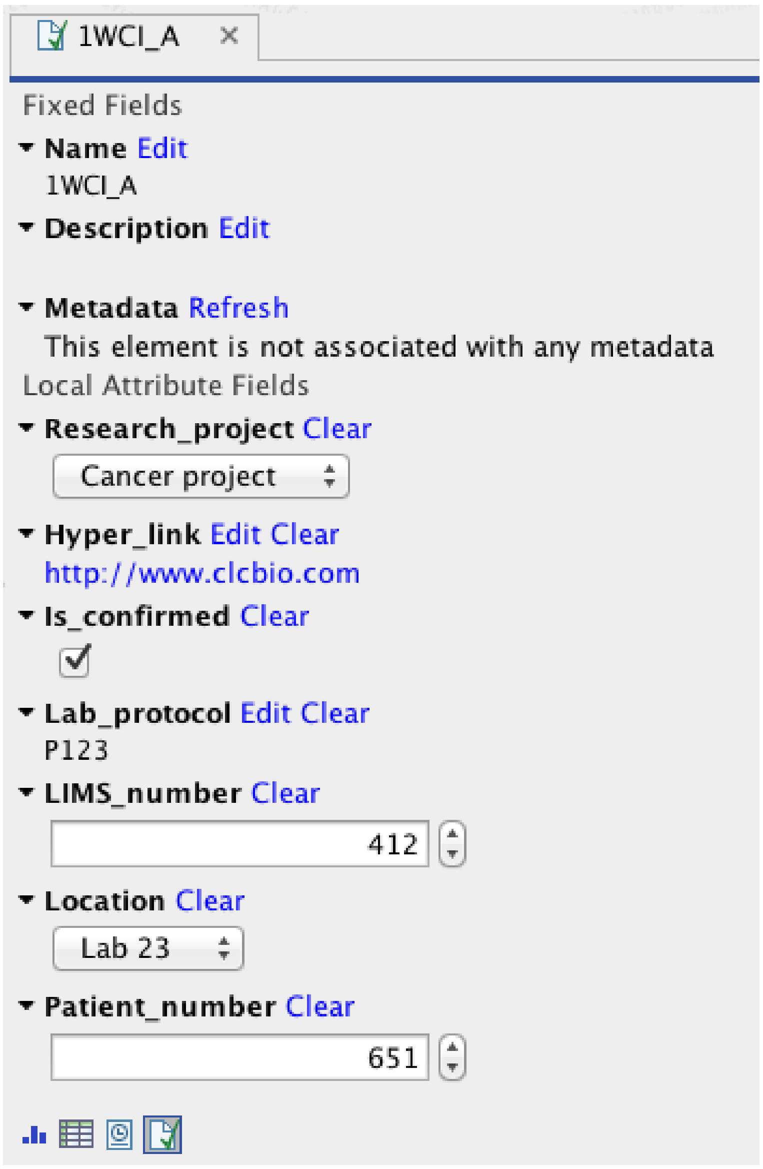
If the element (e.g. sequence) is moved back to the original data location, the information will again be editable and searchable.

If the e.g. Molecule Project or Molecule Table is moved back to the original data location, the information will again be editable and searchable.

5.3.7 Searching

When an attribute has been created, it will automatically be available for searching. This means that in the **Local Search** (), you can select the attribute in the list of search criteria (see figure 5.13).

It will also be available in the **Quick Search** below the **Navigation Area** (press Shift+F1


1WCI_A

Fixed Fields

Name Edit
1WCI_A

Description Edit

Metadata Refresh
This element is not associated with any metadata

Local Attribute Fields

Research_project Clear
Cancer project

Hyper_link Edit Clear
http://www.clcbio.com

Is_confirmed Clear
✓

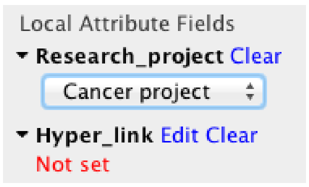
Lab_protocol Edit Clear
P123

LIMS_number Clear
412

Location Clear
Lab 23

Patient_number Clear
651

Figure 5.11: *Adding values to the attributes.*


Local Attribute Fields

Research_project Clear
Cancer project

Hyper_link Edit Clear
Not set

Figure 5.12: *An attribute which has not been set.*

(Fn+Shift+F1 on Mac) and it will be listed - see figure 5.14).

Read more about search in one of the Workbench manuals: http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Local_search.html.

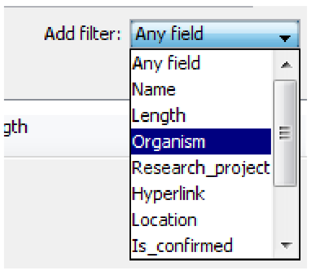


Figure 5.13: *The attributes from figure 5.10 are now listed in the search filter.*

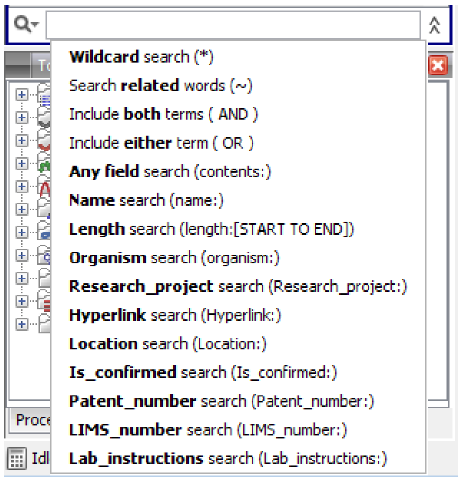

Wildcard search (*)
Search **related** words (~)
Include **both** terms (AND)
Include **either** term (OR)
Any field search (contents:)
Name search (name:)
Length search (length:[START TO END])
Organism search (organism:)
Research_project search (Research_project:)
Hyperlink search (Hyperlink:)
Location search (Location:)
Is_confirmed search (Is_confirmed:)
Patent_number search (Patent_number:)
LIMS_number search (LIMS_number:)
Lab_instructions search (Lab_instructions:)

Figure 5.14: *The attributes from figure 5.10 are now available in the Quick Search as well.*

Chapter 6

Job distribution

The *CLC Server* has the concept of *distributing jobs to nodes*. This means having a master server with the primary purpose of handling user access, serving data to users and starting jobs, and one or more nodes, which execute the jobs submitted to them. This chapter describes the server setups that are available for the *CLC Server* as well as some job running options available for single servers and those running CLC job nodes.

6.1 Introduction to servers setups

The three models for running the *CLC Server* are:

- **Model I: Master server with dedicated job nodes.** In this model, a master server submits CLC jobs directly to machines running the *CLC Server* for execution. In this setup, a group of machines (from two upwards) have the *CLC Server* software installed on them. The system administrator assigns one of them as the *master node*. The master controls the queue and distribution of jobs and compute resources. The other nodes are *job nodes*, which execute the computational tasks they are assigned. This model is 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 Server*. Further details about this setup can be found in section 6.2
- **Model II: Master server submitting to grid nodes.** In this model, a master server submits tasks to a local third party 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 6.3.
- **Model III: Single Server setup**. In this model, the master and execution node functionality is carried out by a single CLC Server instance.

Figure 6.1 shows a schematic overview.

For models I and II, the master server and job nodes, or master server and grid nodes must run on the same type of operating system. It is not possible to have a master server running Linux and a job node running Windows, for example.

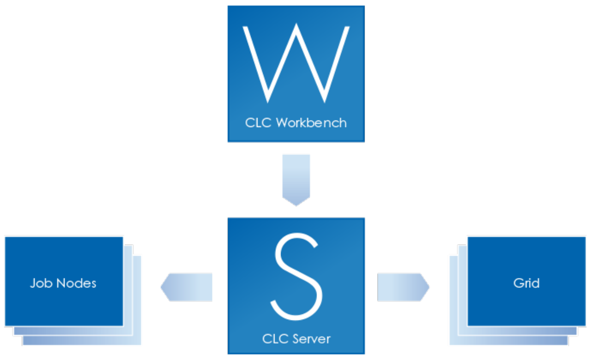


Figure 6.1: *An overview of the job distribution possibilities.*

6.2 Model I: Master server with dedicated job nodes

6.2.1 Overview: Model I

This setup consists of two types of CLC Server instances:

1. **A master node** - a machine that accepts jobs from users and then passes them to job nodes for execution.
2. **Job nodes** - machines running the *CLC Server* that accept jobs directly from a master node.

The general steps for setting up this model are:

1. Install the *CLC Server* software on all the machines involved. (See section 2.2.)
2. Install the license on the machine that will act as the master node. (See section 2.7.)
3. Start up the *CLC Server* software on the master server. Then start up the software on the job nodes. (See section 2.8.)
4. Log in to the web adminstrative interface for the *CLC Server* of the master node. (See section 9.)
5. Configure the master node, attach the job nodes and configure the job nodes via the administrative interface on the master node.

Almost all configuration for the CLC Server cluster is done via the web adminstrative interface for the *CLC Server* on the **master node**. This includes the installation of plugins. See section section 6.2.4.

The only work done directly on the machines that will run as job nodes is

- Installation of the *CLC Server* software.
- Starting up the software up on each node.

- The changing of the built-in administrative login credentials under certain circumstances. See section 6.2.2.
- If using a CLC Bioinformatics Database, installing the relevant database driver on each job node.

6.2.2 User credentials on a master-job node setup

When initially installed, all instances of the *CLC Server* will have the default admin user credentials.

If you have a brand new installation, and you are happy to use the default administrative login credentials (see section 9) during intial setup, you do not need to change anything.

Once the *CLC Server* software on all machines is up and running and the job nodes have been attached to the master, changes to passwords for the built-in authentications system, which includes the default admin user, root, will be pushed from the master to the job nodes. You do not need to manually change the password on each job node.

If you wish to change the default administrative password before attaching the job nodes to the master, then please log into the web administrative interface of each maching running the *CLC Server* software and setup *identical* details on each one.

The master node needs access to the job nodes to be able to push configurations to them. Thus, if you change the admin password on the master server and later wish to attach a new job node, you will need to log into the web administrative interface of the job node and set the root password for the *CLC Server* software to match that of the master server. Until that is done, the master will not be able to communcaate with the job node because the root admin passwords are different. Once the master can communicate with the job node, it can push other configurations to the job node.

6.2.3 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 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:

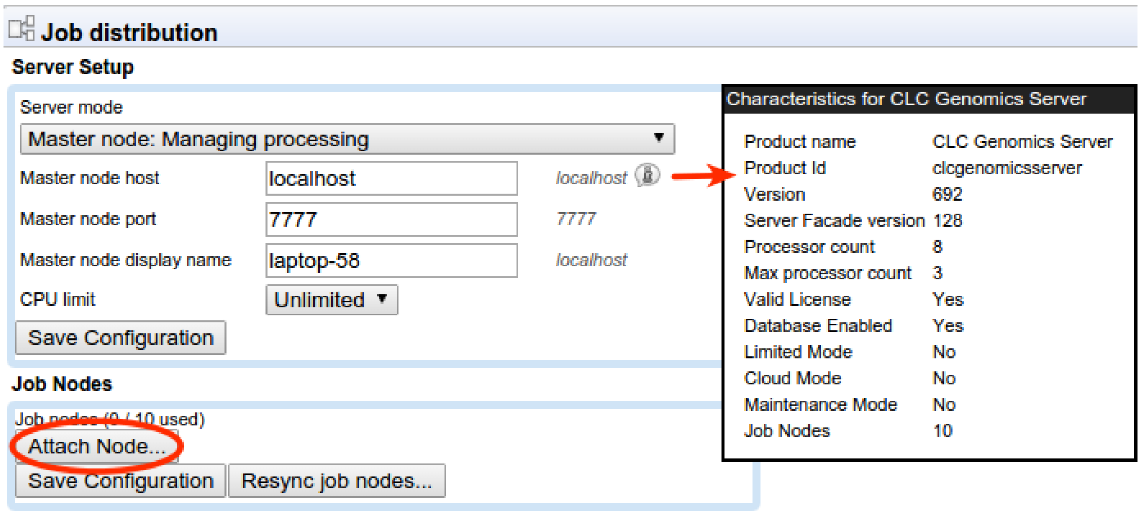
> **Admin ( ) | Job distribution ( )**

First, set the server mode to `MASTER_NODE` and provide the master node address, port and a master node name as shown in figure 6.2. It is optional whether you wish to specify a CPU limit or just leave the field setting to "Unlimited".

The info function next to the "Master node host" field can be used to get information about the server. Clicking the text next to the input text fields will use this text to populate the text fields.

The display name is shown in the top graphics of the administration interface.

If the **Attach Node** button in the Job nodes section is greyed out, click on the button labeled **Save Configuration** in the Server mode section to actively save the `MASTER_NODE` setting. The **Attach Node** button should now be active. Click on it to specify a job node to attach. Fill in the appropriate information about the node (see figure 6.3).



Job distribution

Server Setup

Server mode	Master node: Managing processing	
Master node host	localhost	*localhost*
Master node port	7777	*7777*
Master node display name	laptop-58	*localhost*
CPU limit	Unlimited	

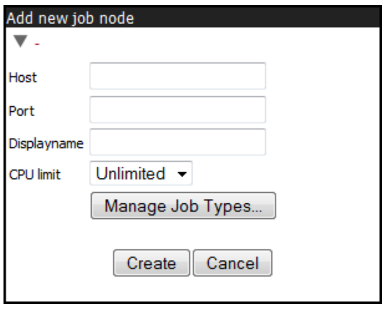
Characteristics for CLC Genomics Server

Product name	CLC Genomics Server
Product Id	clcgenomicsserver
Version	692
Server Facade version	128
Processor count	8
Max processor count	3
Valid License	Yes
Database Enabled	Yes
Limited Mode	No
Cloud Mode	No
Maintenance Mode	No
Job Nodes	10

Job Nodes

Job nodes (0 / 10 used)

Figure 6.2: *Setting up a master server.*



Add new job node

Host	
Port	
Displayname	
CPU limit	Unlimited

Figure 6.3: *Add new job node.*

Besides information about the node hostname, port, displayname, and CPU limit, you can also configure what kind of jobs each node should be able to execute. This is done by clicking the "Manage Job Types" button (see figure 6.3). As shown in figure 6.4 you can now specify whether the job node should run "Any installed Server Command" or "Only selected Server Commands". Clicking "Only selected Server Commands" enables a search field and a list of all server command names and types. The search field can be used to narrow down the server commands by name or type.

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

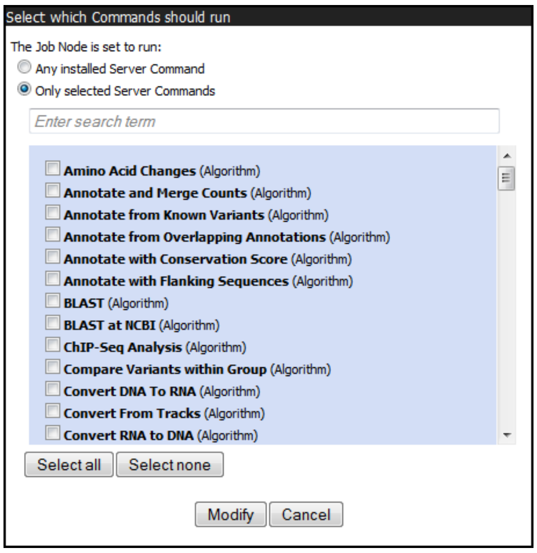
You will get a warning dialog if there are types of jobs that are not enabled on any of the nodes.

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

To test that access works for both job nodes and the master node, you can click "check setup" in the upper right corner as described in section 15.2.1.

One relatively common problem that can arise 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`.

Once set up, job nodes automatically inherit all configurations on the master node. If one of


Select which Commands should run

The Job Node is set to run:

( ) Any installed Server Command
(●) Only selected Server Commands

Enter search term

- [ ] **Amino Acid Changes** (Algorithm)
- [ ] **Annotate and Merge Counts** (Algorithm)
- [ ] **Annotate from Known Variants** (Algorithm)
- [ ] **Annotate from Overlapping Annotations** (Algorithm)
- [ ] **Annotate with Conservation Score** (Algorithm)
- [ ] **Annotate with Flanking Sequences** (Algorithm)
- [ ] **BLAST** (Algorithm)
- [ ] **BLAST at NCBI** (Algorithm)
- [ ] **ChIP-Seq Analysis** (Algorithm)
- [ ] **Compare Variants within Group** (Algorithm)
- [ ] **Convert DNA To RNA** (Algorithm)
- [ ] **Convert From Tracks** (Algorithm)
- [ ] **Convert RNA to DNA** (Algorithm)

Figure 6.4: *Select Server Commands to run on the job node.*

the job nodes gets out of sync with the master, click the **Resync job nodes** button, shown in figure 6.2. This should not be done while jobs are running on any nodes. The server can be put into Maintenance Mode to allow current jobs to complete before maintenance tasks are carried out. See section 7.3.

6.2.4 Installing Server plugins

Server plugin installation is described in section 10.

You **only** need to install the plugin 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.

6.3 Model II: Master server submitting to grid nodes

6.3.1 Overview: Model II

The CLC Grid Integration 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.

A given CLC algorithm will only run on a single machine, i.e., a single job will run on one node. Each grid node employed for a given task must have enough memory and space to carry out that entire task.

The grid system uses two locations for its deployment:

- **Path to CLC Grid Worker** in the grid preset editor. This location is used for plugins, a grid version of the server, i.e., the code that is executed when a grid job is started, licences, libraries and more. Grid workers are redeployed in two situations: 1) When the server starts up and 2) If the configuration of one of the grid presets is changed, in which case the grid workers of all presets are redeployed. In addition, the grid workers are updated when a plugin is installed or removed.

- **Shared work directory**. This location is where each grid job gets its own sub directory in which it places its temporary data, e.g., the description of the job to be executed, the configurations files that the grid version of the server needs in order to setup persistence models and log files. This location is only deployed once, either when the grid job starts executing (in case of workflow jobs) or when the grid job is queued (in all other cases).

6.3.2 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. See Appendix section 15.6 for more information about DRMAA libraries for the supported grid submission systems.
- The *CLC Server* must be installed on a Linux based system configured as a *submit host* in the grid environment.
- The user running the *CLC Server* process is seen as the submitter of the grid job, and thus this user must exist on all the grid nodes.
- *CLC Server* file locations holding data that will be used must be mounted with the same path on the grid nodes as on the master *CLC Server* and accessible to the user that runs the *CLC 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 Server* process.
- A *CLC License Server* with one or more available *CLC Grid Worker* licenses must be reachable from the *execution hosts* in the grid setup.
- A SUN/Oracle Java Runtime environment 1.7 update 45 or later must be installed on all execution hosts that will be running CLC Grid Worker jobs.

Supported grid scheduling systems

CLC officially supports the third party scheduling systems OGE, PBS Pro and IBM Platform LSF. We have tested the following versions:

- OGE 6.2u6
- PBS Pro is 11.0
- LSF 8.3 and 9.1

On a more general level:

- The grid integration in the *CLC Server* is done using DRMAA. Integrating with any submission system that provides a working DRMAA library should in theory 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 Grid Worker licenses, excess tasks are held in the queue until licenses are released. In LSF and 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". This is decribed in more detail in section 6.3.6.

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 Grid Worker licenses. So, with PBS Torque, if you had three Grid Worker 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 Grid Worker licenses you have.

6.3.3 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.

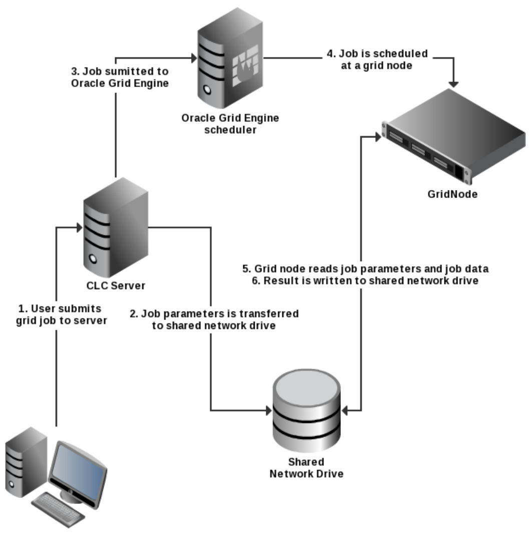

3. Job sumitted to Oracle Grid Engine
4. Job is scheduled at a grid node
Oracle Grid Engine scheduler
GridNode
CLC Server
5. Grid node reads job parameters and job data
6. Result is written to shared network drive
1. User submits grid job to server
2. Job parameters is transferred to shared network drive
Shared Network Drive

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 Server*.

2. The master server writes a file with job parameters to a 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.

6.3.4 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 jobs. These steps are described in more detail in the sections that follow. It is assumed that your *CLC 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 6.3.5
2. Configure the CLC grid licenses as a consumable resource in the local grid system, as described in section 6.3.6
3. Configure and save grid presets, as described in section 6.3.7
4. If not already done, install the CLC Workbench Client Plugin in the Workbenches, as described in section 2.9. The CLC Workbench Client Plugin can 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 6.3.10. 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 Server* client such as the *CLC Server* or the Command Line Tools. Ideally, these would be tasks already known to run smoothly directly on your *CLC Server*.

6.3.5 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://resources.qiagenbioinformatics.com/manuals/clclicenseserver/current/`.

A pdf version is available at

`http://www.resources.qiagenbioinformatics.com//manuals/clclicenseserver/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 are of your *CLC 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 6.3.7, and have saved them, those presets are deployed to the location you specify in the **Path to CLC Grid Worker** 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 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.

6.3.6 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.**

Information about how a consumable resource can be configured for LSF has been provided by IBM and can be found in Appendix section 15.7

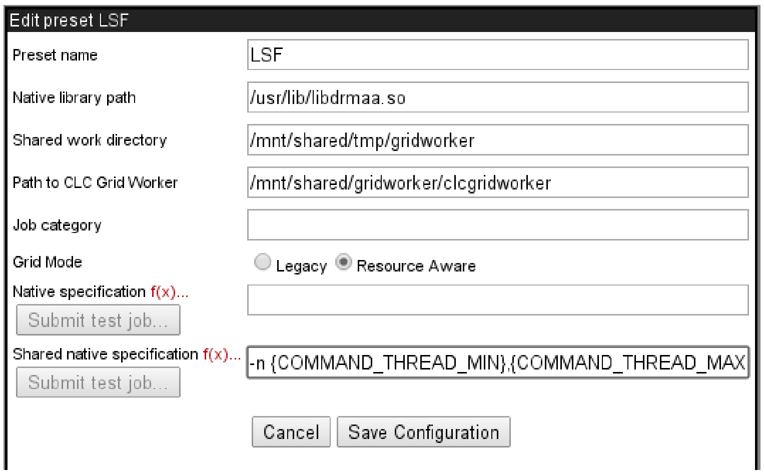
6.3.7 Configure grid presets

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 6.3.12.

To configure the presets, log into the web-interface of the *CLC 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.


Edit preset LSF

Preset name	LSF
Native library path	/usr/lib/libdrmaa.so
Shared work directory	/mnt/shared/tmp/gridworker
Path to CLC Grid Worker	/mnt/shared/gridworker/clcgridworker
Job category	
Grid Mode	Legacy [✓ Resource Aware]
Native specification f(x)...	
Shared native specification f(x)...	-n {COMMAND_THREAD_MIN},{COMMAND_THREAD_MAX

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 6.3.12) 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 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 *CLC 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.

Grid mode "Legacy" or "Resource Aware". Legacy mode is default when migrating an existing pre-6.5 server to 6.5 or later. Resource aware allows the grid preset to use the Shared native specification to submit jobs that do not utilize an entire execution node.

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 6.3.7

Shared native specification Native specification used for jobs that can share the execution node with other jobs. This is described further below 6.3.9

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 esentially 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, five 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 Server* command to be executed on the grid by the clcgridworker executable.

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

COMMAND_THREAD_MIN Evaluates to the minimum number of thread required to run the command being submitted. Only valid in Shared native specification.

COMMAND_THREAD_MAX Evaluates to the maximum number of threads supported by the command being submitted. Only valid in Shared native specification.

These variables can be added by the administrator directly into the Native Specification or Shared 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.

Besides the above-mentioned variables, two functions exist for use in native specifications. Functions are invoked with the following syntax: `{#function arg1, arg2, [... argn]}`:

take_lower_of Evaluates to the lowest integer-value of its argument. The arguments are either a constant integer or a variable-name. If an argument is a a string, which is not a variable-name, or if the variable expands to a non-integer, the argument is ignored. For instance `{#take_lower_of 8,4,FOO}` evaluates to 4 and ignores the non-integer, non-variable "FOO" string.

take_higher_of Evaluates to the highest integer-value of its argument. The arguments are either a constant integer or a variable-name. If an argument is a a string, which is not a variable-name, or if the variable expands to a non-integer, the argument is ignored. For instance `{#take_higher_of 8,4,FOO}` evaluates to 8 and ignores the non-integer, non-variable "FOO" string.

6.3.8 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. By default all cores on an execution node will be used.

All cores on an execution node will be used by default (this is not the case for older versions - before 4.01 - of *CLC Genomics Server*). 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.

Resource Aware Grid Presets

If Resource Aware mode is enabled for a grid preset, algorithms can provide hints regarding their core/thread requirements. Enabling Resource Aware mode allows two separate native specification to be used to submit job:

The string entered in "Native Specification" is used for algorithms that require many resources and are therefore viewed as being "exclusive".

"Shared Native Specification" is used for algorithms that are highly CPU bound and which therefore can share an execution node with other algorithms.

To help configure the resource requirements the two placeholders COMMAND_THREAD_MIN and COMMAND_THREAD_MAX are used. They will be substituted at runtime with the actual minimum or maximum number of threads required by the task about to be scheduled on the grid.

Some grids allow threads to be allocated by requesting a range, other grids only allow a fixed number of threads to be requested. In order to support both scenarios, the take_lower_of and take_higher_of functions are provided which can be used in conjunction with the COMMAND_THREAD_MIN and COMMAND_THREAD_MAX placeholders. If you want to request the maximum number of threads supported by the command being executed, but want the value to be bound by 32, because you have no nodes with more than 32 cores, the following expansion can be used as shared native specification: `{#take_lower_of COMMAND_THREAD_MAX, 32}`.

If a grid preset is set to Legacy-mode, all jobs will be submitted using the Native Specification for exclusive jobs.

Configuration of OGE/SGE

1) CPU Core usage when not using parallel environment

By default the CLC 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.

In the CLC 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 *CLC 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.

Configuration of LSF

With LSF the number of cores to use is specified with the `-n` option. It can be used both with a single argument, and with two arguments. If only one argument is given, `-n X`, LSF interprets the request to be fixed. I.e. exactly X cores are requested. If two arguments are provided, `-n X,Y`, they must be separated by a comma, and are interpreted as a range. I.e. allocate between X and Y cores.

6.3.9 Multi-job Processing on grid

As described in section 6.6.1, the CLC Server supports execution of multiple jobs at the same time on a job node or single server. This is also possible in grid setups. By providing resource information when submitting jobs to the grid, the underlying grid scheduler will be able to schedule multiple jobs on the execution nodes when appropriate.

Algorithms can either be exclusive, streaming, or non-exclusive. Exclusive algorithms are optimized to utilize the machine they are running on. They have very high I/O bandwidth, memory, or CPU requirements and therefore do not play well with other algorithms. Streaming algorithms are highly I/O intensive, and running two streaming algorithms on the same machine does not yield any advantages. For grid execution, streaming algorithms are treated as exclusive algorithms. See the list in Appendix section 15.5 for a list of non-exclusive algorithms.

Non-exclusive algorithms expose their CPU or thread usage, and this can be passed on to the grid scheduler when submitting jobs. In order to pass this information to the grid scheduler, the grid preset must be set to Resource Aware mode:

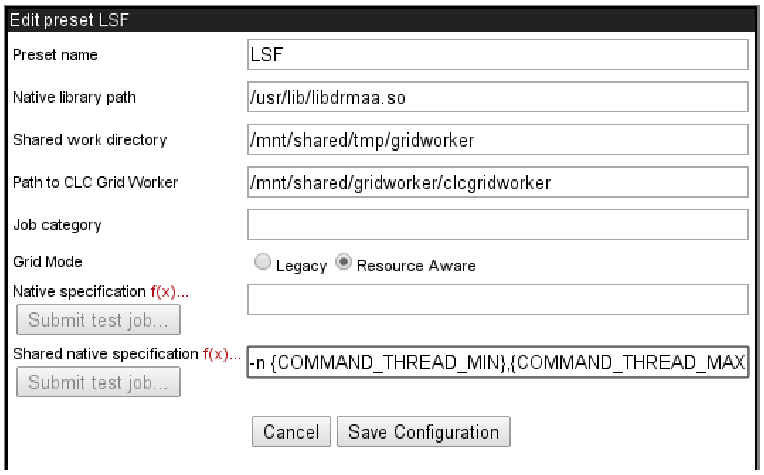
Go to:

Admin ( ) | Job distribution ( )

and click 'Create New Preset...' or 'Edit...' and fill out the fields as described in section 6.3.7 and set 'Grid Mode' to 'Resource Aware'. This will bring up a new field called 'Shared native specification'.

When submitting non-exclusive algorithms the Shared native specification field is used as submission argument. Exclusive algorithms are submitted using the regular native specification.

The Shared native specification must be filled out such that at least the COMMAND_THREAD_MAX variable is passed properly to the grid (for hints see section 6.3.8).


Edit preset LSF

Preset name	LSF
Native library path	/usr/lib/libdrmaa.so
Shared work directory	/mnt/shared/tmp/gridworker
Path to CLC Grid Worker	/mnt/shared/gridworker/clcgridworker
Job category	
Grid Mode	Legacy [✓ Resource Aware]
Native specification f(x)...	
Shared native specification f(x)...	-n {COMMAND_THREAD_MIN},{COMMAND_THREAD_MAX

Figure 6.7: *Grid preset configured for multi-job processing on LSF.*

Figure 6.7 shows a Resource Aware grid preset setup for LSF.

Each CLC Grid Worker launched, whether it is to run alone on a node or run alongside a job already running on a particular node, will attempt to get a license from the CLC License Server. Once the job is complete, the license will be returned.

6.3.10 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 Grid Worker** 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 Server* java process and a temporary directory available from the grid nodes, overriding the defaults that would otherwise apply:

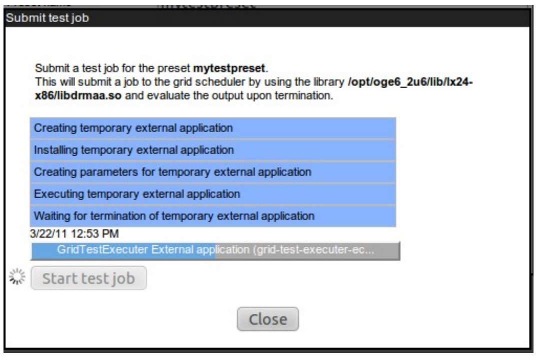
```
-Xmx1000m
-Djava.io.tmpdir=/path/to/tmp
```

For each grid preset you created, you can create a `clcgridworker.vmoptions` file within the folder you specified in the **Path to CLC Grid Worker** field. So for example, if you had two grid presets, you could set two quite different memory limits for the *CLC 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 import jobs and trimming jobs in the case of *CLC Genomics Server*, and one for tasks with higher overheads, such as de novo assemblies or read mappings in the case of *CLC Genomics Server*.

6.3.11 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 6.8.



Submit a test job for the preset **mytestpreset**.
This will submit a job to the grid scheduler by using the library **/opt/oge6_2u6/lib/lx24-x86/libdrmaa.so** and evaluate the output upon termination.

Creating temporary external application
Installing temporary external application
Creating parameters for temporary external application
Executing temporary external application
Waiting for termination of temporary external application
3/22/11 12:53 PM
GridTestExecuter External application (grid-test-executer-ec...

Figure 6.8: *Testing a Grid Preset.*

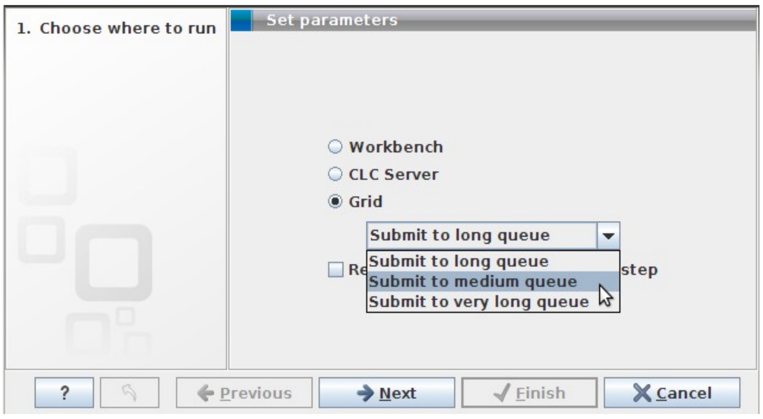
6.3.12 Client-side: starting CLC jobs on the grid

Installing the CLC Workbench Client Plugin

To submit jobs to the grid from within a CLC Workbench, users must be able to access the *CLC Server*, which means that the CLC Workbench Client Plugin 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 Plugin has been installed in the CLC Workbench, an extra option will appear in the first dialog box presented when setting up a task that could be executed on the *CLC 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 (figure 6.9).



1. Choose where to run

Set parameters

Workbench

CLC Server

[✓ Grid]

Submit to long queue

Submit to long queue
Submit to medium queue
Submit to very long queue

Figure 6.9: *Starting the job on the grid.*

6.3.13 Grid Integration Tips

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

- Does your system meets the requirements of the CLC Grid Integration tool 6.3.2? For example, please check that the machine the *CLC Server* is running on is configured as a submit host for your grid system, and please check that you are running Sun/Oracle Java 1.7 on all execution hosts.
- The user running the *CLC 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 Server* file locations mounted with the same path on the grid nodes as on the master *CLC Server* and accessible to the user that runs the *CLC 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 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 seeting up the Database Location using the *CLC 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 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 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 clcgrid-worker script in the *CLC 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 Grid Worker** 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 *CLC Serverr* 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.

6.3.14 Understanding memory settings

Most work done by the *CLC Server* is done via its java process.

However, some tools involving de novo assembly or mapping phases (e.g. read mappings, RNA-seq analyses, smallRNA analyses, etc.) on *CLC Genomics Server* use external native 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 6.3.10.

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 1.2 GB.

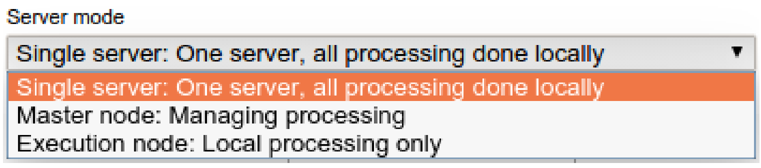
External binaries

For the computationally intensive tools that include a phase using an external binary, the 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.

6.4 Model III: Single Server setup

In this model, the master and execution node functionality is carried out by a single CLC Server instance. Here, the *CLC Server* software is installed on a single machine. Jobs submitted to the server are executed on this same machine.

To designate the system as a single server, after installation and starting the server, select the option SINGLE_SERVER from the drop down list of Server modes.



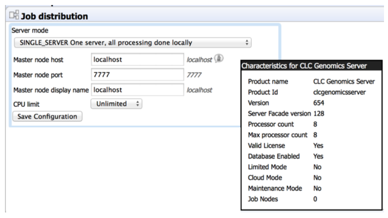
Server mode

Single server: One server, all processing done locally ▼

Single server: One server, all processing done locally
Master node: Managing processing
Execution node: Local processing only

Figure 6.10: *The configuration options for the types of machines running the* **CLC 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 Node option. This option is there primarily to allow for troubleshooting.*

You can then configure node hostname (usually localhost), port (usually 7777), displayname, and CPU limit (figure 6.11). The info function next to the "Master node host" field can be used to get information about the server. Clicking the text next to the input text fields will use this text to populate the text fields. It is optional whether you wish to specify a CPU limit or just leave the field setting to "Unlimited".



Job distribution

Server mode: SINGLE_SERVER One server, all processing done locally

Master node host	localhost	*localhost*
Master node port	7777	*7777*
Master node display name	localhost	*localhost*
CPU limit	Unlimited	

Save Configuration

Characteristics for CLC Genomics Server

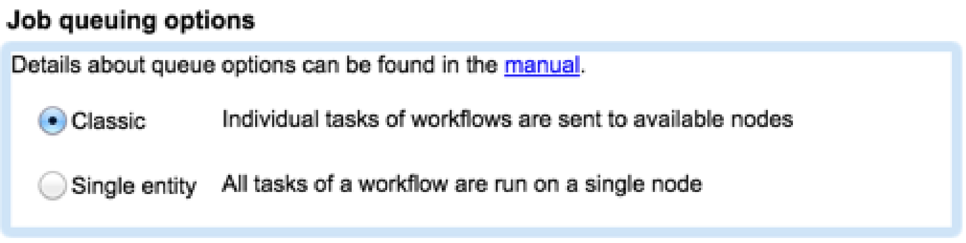
Product name	CLC Genomics Server
Product Id	clcgenomicsserver
Version	654
Server Facade version	128
Processor count	8
Max processor count	8
Valid License	Yes
Database Enabled	Yes
Limited Mode	No
Cloud Mode	No
Maintenance Mode	No
Job Nodes	0

Figure 6.11: *Add new job node.*

For more information about the maximum amount of jobs that can be concurrently run on a single server (10), see section 6.6.3.

6.5 Job queuing options

The way Workflows should be handled on server setups with nodes can be configured using the Job queuing options found under the Job distribution tab (figure 6.12).



Job queuing options

Details about queue options can be found in the manual.

[✓ Classic] Individual tasks of workflows are sent to available nodes

Single entity All tasks of a workflow are run on a single node

Figure 6.12: *Job queuing options are available that determine how Workflow jobs are handled.*

The options are:

- **Classic** Each task of a Workflow is scheduled separately for execution. For example, a Workflow with 10 tasks would result in 10 jobs being submitted. Each of those jobs can be sent to any available node with adequate resources when that step is ready to be run. Classic is the default setting.

- **Single entity** A single job submission is made for a Workflow, regardless of how many tasks that Workflow consists of. All tasks of that Workflow are run on the same node.

6.5.1 Choosing between Classic and Single entity options

Choosing the best queuing option involves consideration of the types of analyses being run and the system the jobs are being run on. The following are some considerations to help guide this decision.

Classic

The Classic option would be beneficial in the following situations:

- **Workflows containing parallel branches are commonly launched.** Elements of such Workflows can be scheduled on multiple nodes. This potential for parallel execution of Workflow steps can yield shorter average running times, depending on other aspects of the setup (e.g. node or grid license availability). Time savings are most noticeable on systems with spare capacity when running Workflows with computationally intensive elements on parallel branches.
- **Single tools are predominantly submitted**, as opposed to Workflows. There is no need to change the default setting in this case, as the Single entity option only affects the way Workflows are handled.
- **Job node setups only: Nodes have been dedicated to certain types of analyses** If a job node has been configured to run only certain tasks (Server commands) as described in section 6.2.3, then, with the Classic option, this node can be used for those configured tasks, whether or not they are elements of a Workflow. This is not the case with the Single entity option where Workflows cannot be run on the job nodes configured like this.

Single entity

The Single entity option would be beneficial when the execution of Workflows is common, and:

- **Workflows being submitted often consist of large number of steps and each step has small computational requirements.** This is common when working with data from organisms with small genomes, such as bacterial or viral samples, or when working with enriched data from organisms with large genomes. Where scheduling overhead outweighs the net analysis time of a given Workflow step, the Single entity setting can yield many-fold improvements in overall sample throughput.
- **Nodes frequently run at capacity.** By running an entire Workflow on a single node, the Single entity option can leverage caching mechanisms, aiding performance. When all nodes are busy much of the time, the opportunity for gains through concurrent processing of parallel Workflow branches on multiple nodes is much lower than on a system with spare capacity. On such a system, the Single entity option may thus yield overall performance gains, even when running Workflows containing elements with computationally intensive steps.
- **The node hardware is homogeneous.** All the nodes should be of a size that could handle all tasks in the Workflows being submitted.

- **Resource allocation is a focus.** On setups where many users are sharing the resources and are running Workflows, the Single entity option may help with resource access for different users. For example, consider a grid node setup with 20 nodes, where one user submits 15 Workflows with 10 tasks in each Workflow. With the Classic option, this would lead to 150 jobs, which can be sent across all 20 nodes. When the next user submits a job, it would be queued behind all of those 150 jobs from the first user. With the Single entity option, the 15 Workflows would have been submitted to 15 nodes, leaving 5 nodes available on which the other user's job could be run.

- **Large numbers of Workflows are submitted during limited periods of time, each Workflow consisting of several or many tasks.** Such a situation leads to thousands of jobs in the queue using the Classic option. Using the Single entity option is a way to keep the scheduling load on the master node with reasonable limits.

- **Grid node setups only: The number of grid worker licenses is limited relative to the number of job submissions.** Using the Single entity option, a Workflow is submitted as a single job and thus consumes a single grid worker license. If a Workflow has 10 steps and is submitted using the classic option, 10 jobs are created. Each of these jobs will consume a license, making license availability a limiting factor, along with node availability, for when jobs can be run.

- **Grid node setups only: Resource tracking is a focus.** Workflows involving many steps would run as a single job with a single license consumed by that job. This potentially decreases the complexity of resource use tracking of users running Workflows.

6.6 Job running options

Some jobs can run alongside others on a CLC Server in single server mode (Model III) or on a job node in a CLC Server job node setup (Model I). This feature can be enabled, disabled and configured within the section of the web administrative interface:

Admin ( ) | Job distribution ( ) | Job running options

These settings do not have relevance for grid nodes. Please see section 6.3.9 for more information about concurrent job execution on grid nodes.

There are three categories of jobs relevant to this feature.

- **Non-exclusive algorithms** These are analyses that can run at the same time as others in this category as well as the streaming category, described below. They have low demands on system resources. An example of a non-exclusive algorithm would be "Convert from tracks".

- **Stream algorithms** These have high I/O demands, that is, they have high demands for reading from and writing to disk. Such jobs cannot be run with others in the streaming category but can be run alongside jobs in the non-exclusive category. An example of a streaming job would be import of NGS data.

- **Exclusive algorithms** These are algorithms optimized to utilize the machine they are running on. They have very high I/O bandwidth, memory, or CPU requirements and therefore do not play well with other algorithms. An example of this sort of analysis would be "Map Reads to Reference".

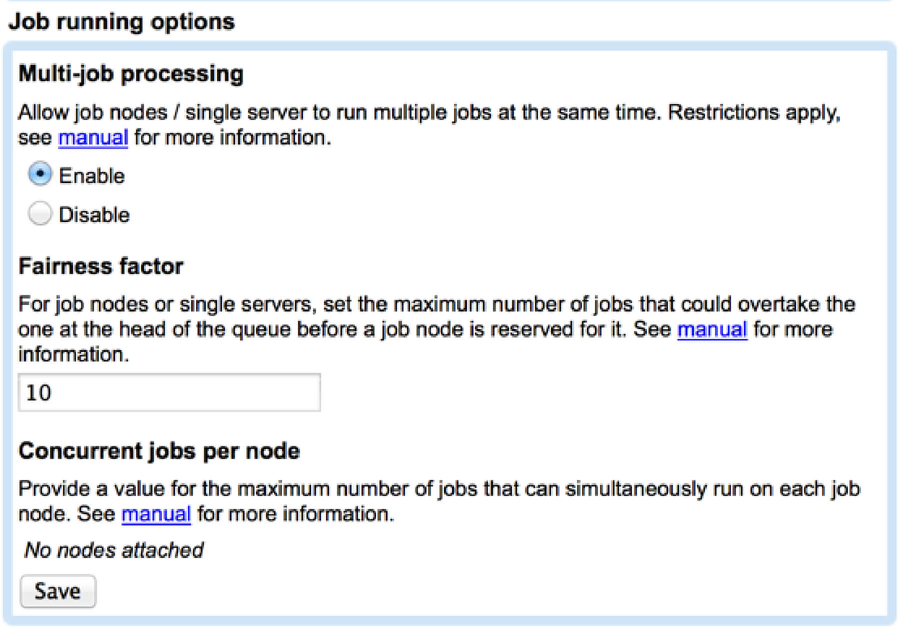
See Appendix section 15.5 for a list of *CLC Genomics Server* algorithms that can be executed concurrently on on a job node or single server.

The rest of this section discusses the available configuration options, as they apply to the running of multiple non-exclusive jobs, or running a single streaming job alongside non-exclusive jobs, on a CLC Server in single server mode or on a job node.

6.6.1 Multi-job processing

Allowing more than one analysis to run on a CLC Server in single server mode or on a job node is enabled by default. This feature can be disabled in the **Multi-job processing** section of the interface by setting the "Multi-job Processing" option to "Disable" (figure 6.13). Click on the button labeled "Save" to save this change.

When this feature is disabled all jobs will be executed one at a time.


Job running options

Multi-job processing

Allow job nodes / single server to run multiple jobs at the same time. Restrictions apply, see manual for more information.

[✓ Enable] Disable

Fairness factor

For job nodes or single servers, set the maximum number of jobs that could overtake the one at the head of the queue before a job node is reserved for it. See manual for more information.

10

Concurrent jobs per node

Provide a value for the maximum number of jobs that can simultaneously run on each job node. See manual for more information.

No nodes attached

Figure 6.13: *The default status is to enable Multi-job processing. Select "Disable" to require that only a single job is executed at a given time on a job node or single server.*

6.6.2 Fairness factor

The fairness factor defines the number of times that a job in the queue can be overtaken by other jobs before resources are reserved for it to run. So, for example, in a situation where there are many non-exclusive jobs and some exclusive jobs being submitted, it is desirable to be able to clear the queue at some point to allow the exclusive job to have a system to itself so it can run. The fairness factor setting is used to determine how many jobs can move ahead of an exclusive job in the queue before the exclusive job will get priority and a system will be reserved for it. The same fairness factor applies to streaming jobs being overtaken in the queue by non-exclusive jobs.

The default value for this setting is 10. With this value set, a job could be overtaken by 10 others before resources are reserved for it that will allow it to run. A fairness factor of 0 means that a node will be reserved for the job at the head of the queue.

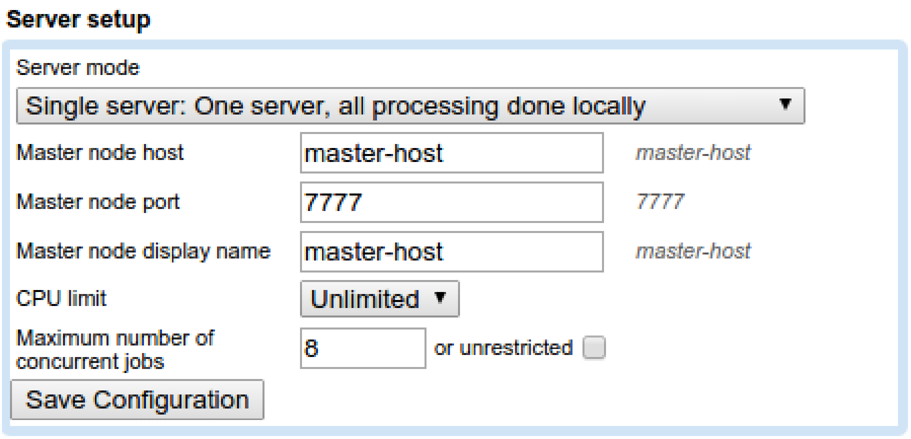
This value can be configured under the **Fairness factor** section of:

> **Admin ( ) | Job distribution ( ) | Job queuing options**

6.6.3 Concurrent jobs per node

By default, the maximum number of jobs that can be concurrently run on a single server or node is 10. This number can be configured on a per-system basis. The maximum number of concurrent jobs that can be configured is set to the number of cores on the node.

Single server setup Go to the section Server Setup under the Job Distribution tab of the web administration page and enter the desired value in the box labeled "Maximum number of concurrent jobs" (figure 6.14).

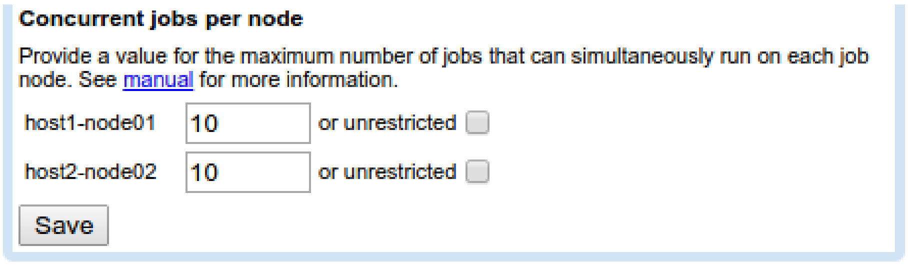

Server setup

Server mode: Single server: One server, all processing done locally

Master node host	master-host	master-host
Master node port	7777	7777
Master node display name	master-host	master-host
CPU limit	Unlimited	
Maximum number of concurrent jobs	8	or unrestricted ☐

Figure 6.14: *Set the maximum number of concurrent jobs or check the "Unrestricted" box to change the setting for the single server setup.*

Job node setup Go to the section Job queuing options under the Job Distribution tab of the web administration page and enter a value for each job node listed under the "Concurrent jobs per node" section (figure 6.15).


Concurrent jobs per node

Provide a value for the maximum number of jobs that can simultaneously run on each job node. See manual for more information.

host1-node01	10	or unrestricted ☐
host2-node02	10	or unrestricted ☐

Figure 6.15: *Set the maximum number of concurrent jobs or check the "Unrestricted" box for any job node.*

Chapter 7

Status and management

Server operation can be managed from the Admin tab, under Status and Management (figure 7.1).

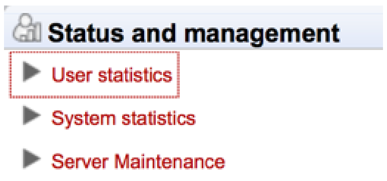
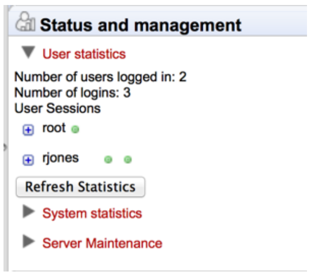


Figure 7.1: *The Status and management tab.*

7.1 User statistics

The User statistics section contains information about the number of users logged in, the number of active sessions (number of logins), and information about each active session. An example is shown in figure 7.2.

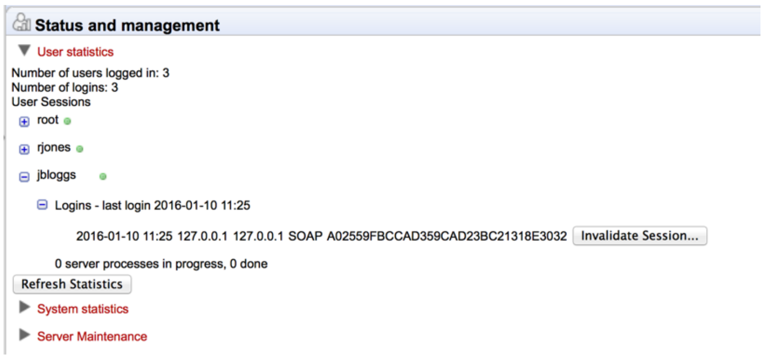


Number of users logged in: 2
Number of logins: 3
User Sessions
root ●
rjones ● ●

Figure 7.2: *Information about the number of users and active sessions (logins) is provided in the User statistics area. Here, two users are logged in. rjones has two active sessions and root has one.*

A green dot by a user's name indicates that they are logged into the server. Two green dots indicate that this user is logged in twice. For example, perhaps they have Workbenches running on two different systems and are logged in via both Workbenches. A grey dot means they have previously logged in but are not at this time.

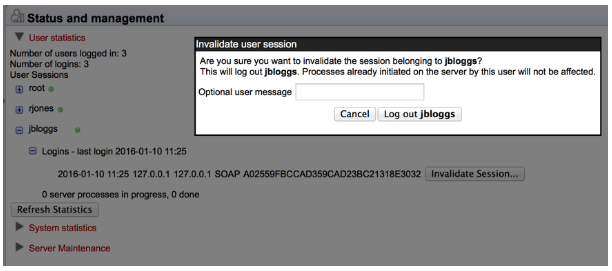
Click on the small button with a plus to the left of a username to expand the information about that user's sessions (figure 7.3). You can also log users off the server by clicking on the **Invalidate Session...** button. This opens a confirmation dialog where a message to the user can be written. This message is displayed via the user's active session. For example, if they are logged into a Workbench, a window will pop up saying they have been logged out of the server and also containing the message written in this field. This action forcibly logs the user out of the *CLC Server*. This action does **not** stop jobs already submitted or running on the server. Optionally, you can send a message to the user whose session is being terminated (see figure 7.4). If the user is logged into the *CLC Server* from a CLC Workbench, then the message entered will appear in a warning box that pops up via the CLC Workbench.



Number of users logged in: 3
Number of logins: 3
User Sessions
root
rjones
jbloggs
Logins - last login 2016-01-10 11:25
2016-01-10 11:25 127.0.0.1 127.0.0.1 SOAP A02559FBCCAD359CAD23BC21318E3032

0 server processes in progress, 0 done

Figure 7.3: *Details about jbloggs' session can be seen by clicking on the small button to the left of that username.*



Number of users logged in: 3
Number of logins: 3
User Sessions
root
rjones
jbloggs
Logins - last login 2016-01-10 11:25
2016-01-10 11:25 127.0.0.1 127.0.0.1 SOAP A02559FBCCAD359CAD23BC21318E3032

0 server processes in progress, 0 done

Are you sure you want to invalidate the session belonging to **jbloggs**?
This will log out **jbloggs**. Processes already initiated on the server by this user will not be affected.
Optional user message

Figure 7.4: *Clicking on the Invalidate Session button will forcibly log a user out of the CLC Server. The admin can optionally provide a message to the user when doing this.*

7.2 System statistics

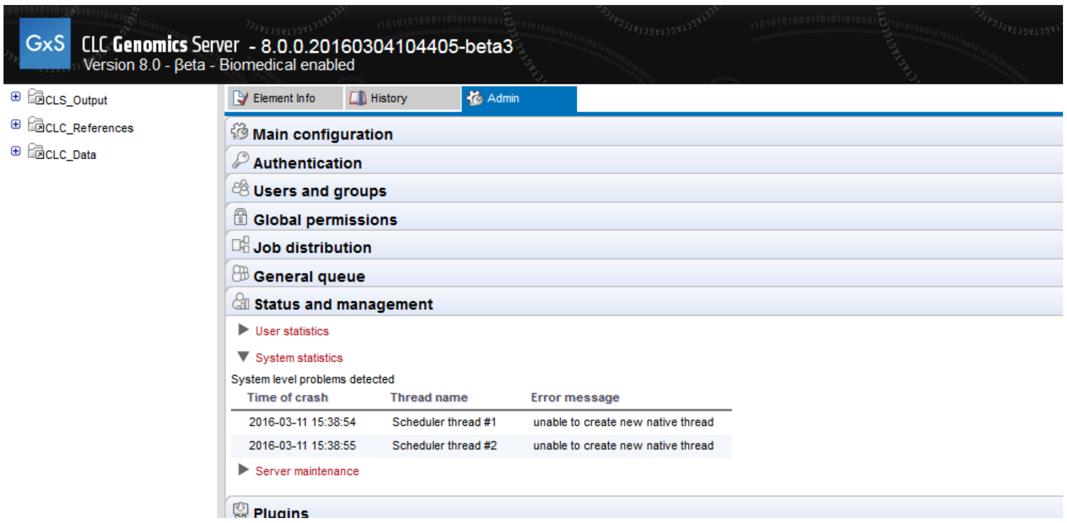
Crashed threads, suggesting system level problems, are reported in this area. In some instances, a system restart may be needed to resolve the issue.

The message "No system level problems detected" is shown in this area if no problems have been detected. An example of the information provided when a problem is detected is shown in figure 7.5. In the case shown, the job submission threads were dead, with the problem reported here and in more detail in the CLC Server log files.

7.3 Server maintenance

Settings under the Server maintenance tab allow a server administrator to change the operating mode of the server and send out messages to users of the *CLC Server* (see figure 7.6).

- **Normal Operation** The *CLC Server* is running.
- **Maintenance Mode** Current jobs are allowed to run and complete, but submission of new jobs is restricted. While the server is in maintenance mode, users already logged



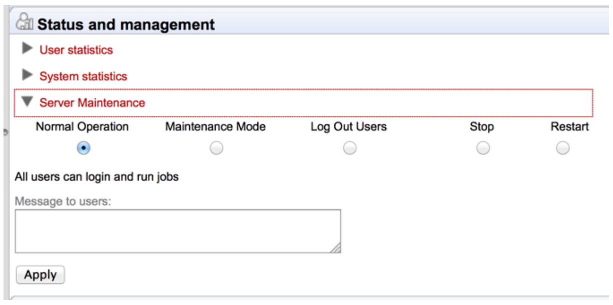
System level problems detected

Time of crash	Thread name	Error message
2016-03-11 15:38:54	Scheduler thread #1	unable to create new native thread
2016-03-11 15:38:55	Scheduler thread #2	unable to create new native thread

Figure 7.5: *System level problems detected and reported in the system statistics area.*

in can check the progress of their jobs or view their data, but they cannot submit new jobs. Users not already logged in cannot log in. An administrator can write a warning message, for example, to inform users about the expected period of time the server will be in maintenance mode.

- **Log Out Users** All users currently logged in will be logged out. All running jobs will be allowed to complete. No users can log in while in this mode. An administrator can also write a warning message for the users.
- **Shut down** The *CLC Server* and any attached job nodes will shut down.
- **Restart** The *CLC Server* and any attached job nodes will be shut down and restarted.



Normal Operation [✓] Maintenance Mode Log Out Users Stop Restart

All users can login and run jobs

Message to users:

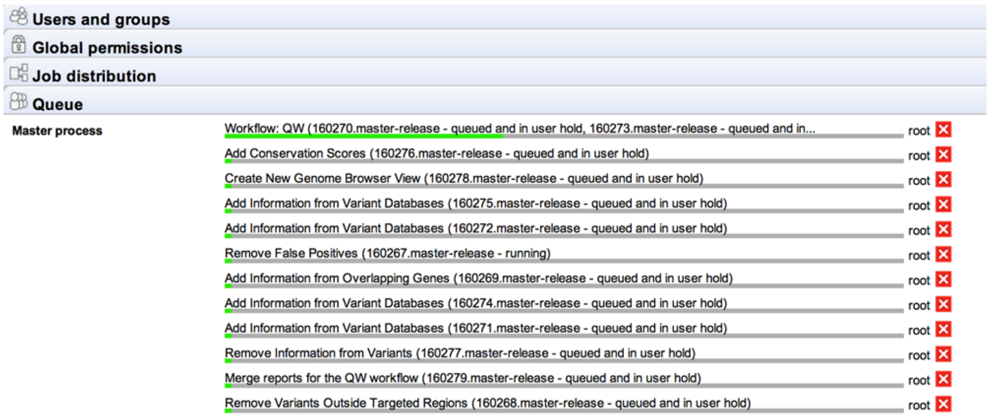
Figure 7.6: *The server administrator can control the operating mode of the CLC Server from under the Server maintenance tab.*

Chapter 8

Queue

Clicking the **Queue** panel will show a list of all the processes that are currently in the queue including jobs in progress.

An example is shown in figure 8.1.



Master process

Workflow: QW (160270.master-release - queued and in user hold, 160273.master-release - queued and in... root
Add Conservation Scores (160276.master-release - queued and in user hold) root
Create New Genome Browser View (160278.master-release - queued and in user hold) root
Add Information from Variant Databases (160275.master-release - queued and in user hold) root
Add Information from Variant Databases (160272.master-release - queued and in user hold) root
Remove False Positives (160267.master-release - running) root
Add Information from Overlapping Genes (160269.master-release - queued and in user hold) root
Add Information from Variant Databases (160274.master-release - queued and in user hold) root
Add Information from Variant Databases (160271.master-release - queued and in user hold) root
Remove Information from Variants (160277.master-release - queued and in user hold) root
Merge reports for the QW workflow (160279.master-release - queued and in user hold) root
Remove Variants Outside Targeted Regions (160268.master-release - queued and in user hold) root

Figure 8.1: *The process queue.*

For each process, you are able to **Cancel** (☐) the processes. At the top, you can see the progress of the process that is currently running.

Chapter 9

Audit log

The audit log records the actions performed on the *CLC Server*. Included are actions like logging in, logging out, import, and the launching and running of analysis tasks. Data management operations such as copying, deleting and adding files are not Server actions and are thus not recorded.

Audit log information is available via the web administrative interface under the **Audit log** tab. The information presented here is stored in a database. Once a month, and when the *CLC Server* is started up, entries in the audit log older than 3 months are deleted.

The limit the audit log database can grow to is 64 GB. If a new entry will push the size past this limit, the system will remove some of the oldest entries so that is is possible for newer entries to be added.

Audit information is also written to text-based log files. Upon the first activity on a given date, a new log file called audit.log is created. This file is then used for logging that activity and subsequent Server activities on that day. When this new audit.log file is created, the file that previously had that name is renamed to audit.<actual events date>.log. These log files are retained for 31 days. When the creation of a new audit.log file is triggered, audit log files older than 31 days are checked for and deleted.

The audit log files can be found under the Server installation area under `webapps/CLCServer/WEB-INF`.

The audit log text files are tab delimited and have the following fields:

- Date and time
- Log level
- Operation: Login, Logout, Command queued, Command done, Command executing, Change server configuration, Server lifecycle; more may be added and existing may be changed or removed.
- Users
- IP Address
- Process name (when operation is one of the Command values) or description of server lifecycle (when operation is Server lifecycle)

- Process identifier - can be used to differentiate several processes of the same type.
- Status - can be used to identify whether the entry was successful or not, e.g. if a job execution failed it will be marked here. Any number other than 0 means failed.

Chapter 10

Server plugins

You can install plugins on the *CLC Server* under the **Admin** ( ) tab (see figure 10.1).

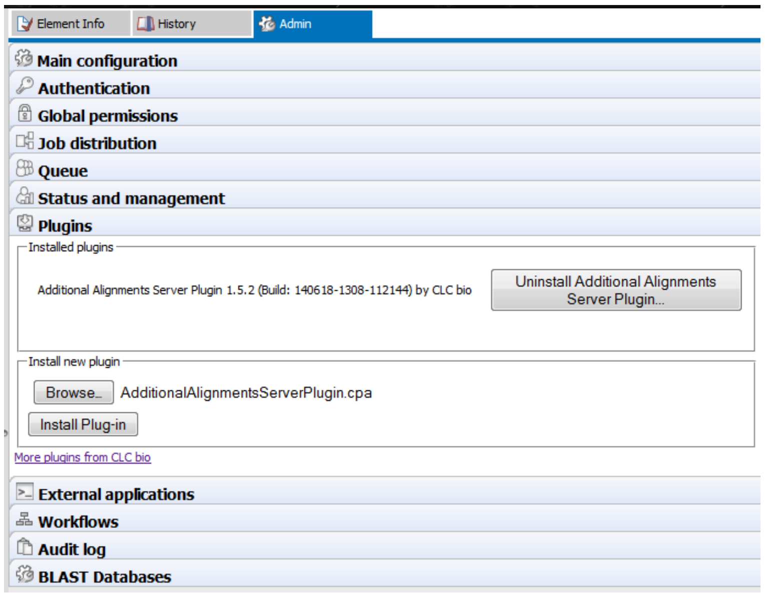


Figure 10.1: *Installing and uninstalling server plugins.*

Click the **Browse** button and locate the plugin .cpa file to install a plugin. To uninstall a plugin, simply click the button next to the plugin.

Installing a plugin will require the *CLC Server* to be restarted. All jobs still in the queue at the time the server is shut down will be dropped and will thus need to be resubmitted after restart. For these reasons, we recommend the use of the Maintenance Mode prior to restarting the *CLC Server*. Maintenance Mode can be found under the Status and management tab (figure 7.1) and is described in section 7. By allowing current jobs to run but no new jobs to be submitted, the impact of a restart on users can be lessened. Once all current jobs have been completed, the *CLC Server* can be restarted on the server and, where relevant, job nodes.

For further information about plugins on job node setups, please refer to section 6.2.4. Grid workers will be re-deployed when the plugin is installed on the master server so no further actions are needed to enable the plugin for use on grid nodes. See section 6.3 for further details about grid worker re-deployment.

Chapter 11

BLAST

The *CLC Server* supports running BLAST jobs submitted from the workbenches that have BLAST tools and from *CLC Server Command Line Tools*. Users will be able to select data from Server **data locations** (see section 3.2.1) to search against other sequences held in Server data locations, or against BLAST databases stored in an area configured as an **import/export directory** (see section 3.3).

11.1 Adding directories for BLAST databases on the Server

In the web interface of the server, you can configure your Server for BLAST databases:

Admin ( ) | BLAST Databases ( )

Here, you can add a folder where you want the Server to look for BLAST databases. However, before doing this, please ensure that the folder you will be adding has been configured as an **import/export directory** (see section 3.3). This is necessary because BLAST databases are not truly CLC data, and thus are stored outside data locations specified for CLC data. They need, however, to be stored somewhere accessible to *CLC Server*process though, hence the need to put them in a directory configured as an import/export directory.

After the folder holding BLAST databases is configured as an Import/Export directory, it can be configured as a location that the *CLC Server* will look in for BLAST databases.

Do this by clicking on the **Edit BLAST Database Locations** button at the bottom of the area under the BLAST Databases area in the administrative interface.

This will bring up a dialog as shown in figure 11.1 where you can select which of the import/export directories you wish to use for storing BLAST databases.

Once added as a BLAST Database Location, the *CLC Server* will search this directory for any BLAST databases and list them under the BLAST tab in the web interface (see a section of this as an example in figure 11.2).

This overview is similar to the one you find in the Workbench BLAST manager for local databases including the following in formation:

- **Name.** The name of the BLAST database.

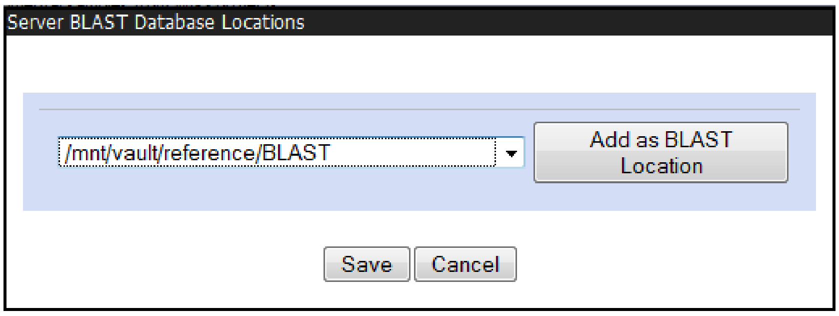
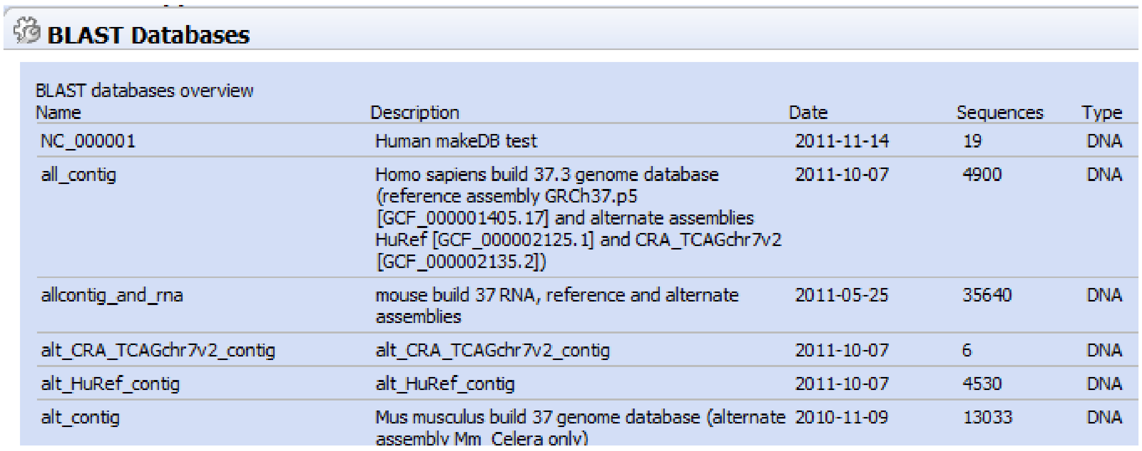


Figure 11.1: *Adding import/export directories as BLAST database locations.*


BLAST Databases

BLAST databases overview

Name	Description	Date	Sequences	Type
NC_000001	Human makeDB test	2011-11-14	19	DNA
all_contig	Homo sapiens build 37.3 genome database (reference assembly GRCh37.p5 [GCF_000001405.17] and alternate assemblies HuRef [GCF_000002125.1] and CRA_TCAGchr7v2 [GCF_000002135.2])	2011-10-07	4900	DNA
allcontig_and_rna	mouse build 37 RNA, reference and alternate assemblies	2011-05-25	35640	DNA
alt_CRA_TCAGchr7v2_contig	alt_CRA_TCAGchr7v2_contig	2011-10-07	6	DNA
alt_HuRef_contig	alt_HuRef_contig	2011-10-07	4530	DNA
alt_contig	Mus musculus build 37 genome database (alternate assembly Mm_Celera only)	2010-11-09	13033	DNA

Figure 11.2: *Selecting database to BLAST against.*

- **Description.** Detailed description of the contents of the database.
- **Date.** The date the database was created.
- **Sequences.** The number of sequences in the database.
- **Type.** The type can be either nucleotide (DNA) or protein.
- **Total size (1000 residues).** The number of residues in the database, either bases or amino acid.
- **Location.** The location of the database.

To the right of the Location information is a link labeled Delete that can be used to delete a BLAST database.

11.2 Adding and removing BLAST databases

Databases can be added in two ways:

- Place pre-formatted databases in the directory selected as BLAST database location on the server file system. The *CLC Server* will automatically detect the database files and list the database as target when running BLAST. You can download pre-formatted database from e.g. ftp://ftp.ncbi.nih.gov/blast/db/.

- Run the **Create BLAST Database ( )** tool via your Workbench, and choose to run the function on the Server when offered the option in the Workbench Wizard. You will get a list of the BLAST database locations that are configured on your Server. The final window of the wizard offers you a location to save the output to. The output referred to is the log file for the BLAST database creation. The BLAST databases themselves are stored in the designated BLAST database folder you chose earlier in the setup process.

A note on permissions: To create BLAST databases on the Server, using the Workbench interface, the user **running the** *CLC Server***process** must have file system level write permission on the import/export directory that you have configured to hold BLAST database.

By default, if you do not change any permissions within *CLC Server*, all users logging into the *CLC Server*(e.g. via their Workbench, or via the Command Line Tools), will be able to create BLAST databases in the areas you have configured to hold BLAST databases.

If you wish to restrict the ability to create BLAST databases to these areas completely, but still wish your users to be able to access the BLAST databases to search against, then set the file system level permissions on the import/export directory so they are read-only.

When listing the databases as shown in figure 11.2, it is possible to delete the databases by clicking the **Delete** link at the far right-hand side of the database information.

Chapter 12

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 Server*. These tools can access data on the machine the CLC Workbench is installed on, data stored on the *CLC Server* or data stored in areas of the server accessible to the *CLC Server*, depending on choices made by the server administrator.

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

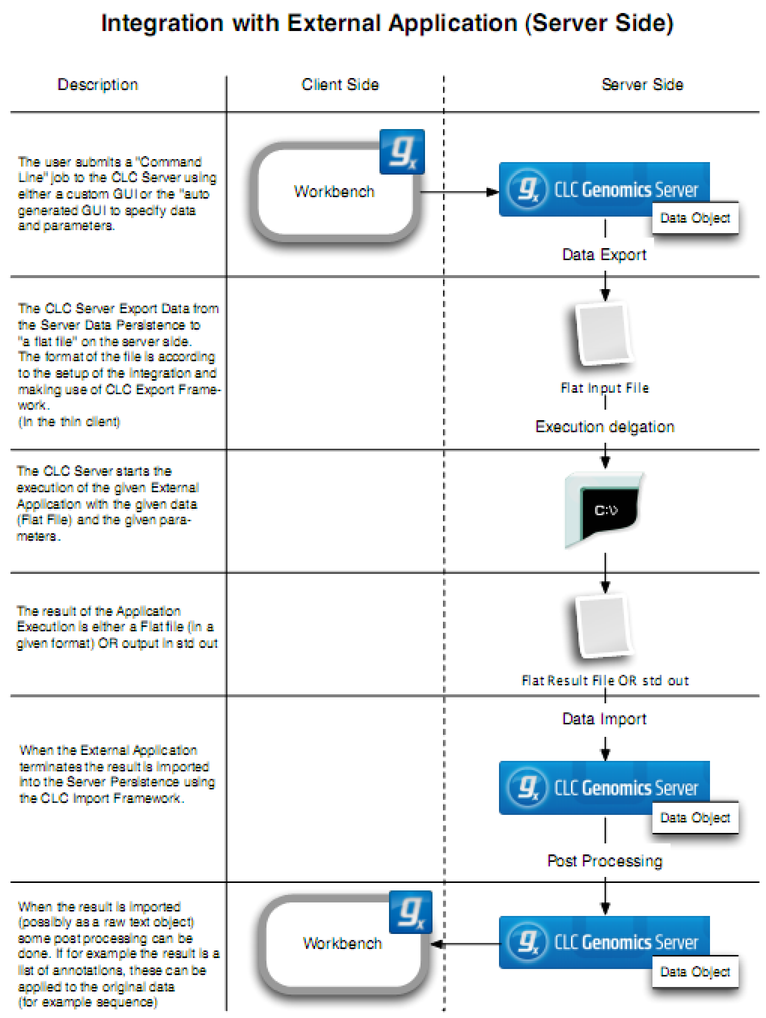
The *External Application Client Plugin* must be installed on the CLC Workbench for access to external applications from the CLC Workbench. This plugin can be found in the Workbench Plugin Manager.

An important note about the execution of External Applications: It is important to consider that, like other tasks executed via the *CLC Server*, any tool configured as an External Application, will be run as the **same logical user** that runs the *CLC Server* process itself. *If you plan to configure 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 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 12.1 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



The user submits a "Command Line" job to the CLC Server using either a custom GUI or the "auto generated GUI to specify data and parameters.

The CLC Server Export Data from the Server Data Persistence to "a flat file" on the server side. The format of the file is according to the setup of the integration and making use of CLC Export Frame-work. (In the thin client)

The CLC Server starts the execution of the given External Application with the given data (Flat File) and the given para-meters.

The result of the Application Execution is either a Flat file (in a given format) OR output in std out

When the External Application terminates the result is imported into the Server Persistence using the CLC Import Framework.

When the result is imported (possibly as a raw text object) some post processing can be done. If for example the result is a list of annotations, these can be applied to the original data (for example sequence)

Figure 12.1: *An overview of the external applications integration in this example illustrated with CLC Genomics Workbench and CLC Genomics Server.*

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

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.

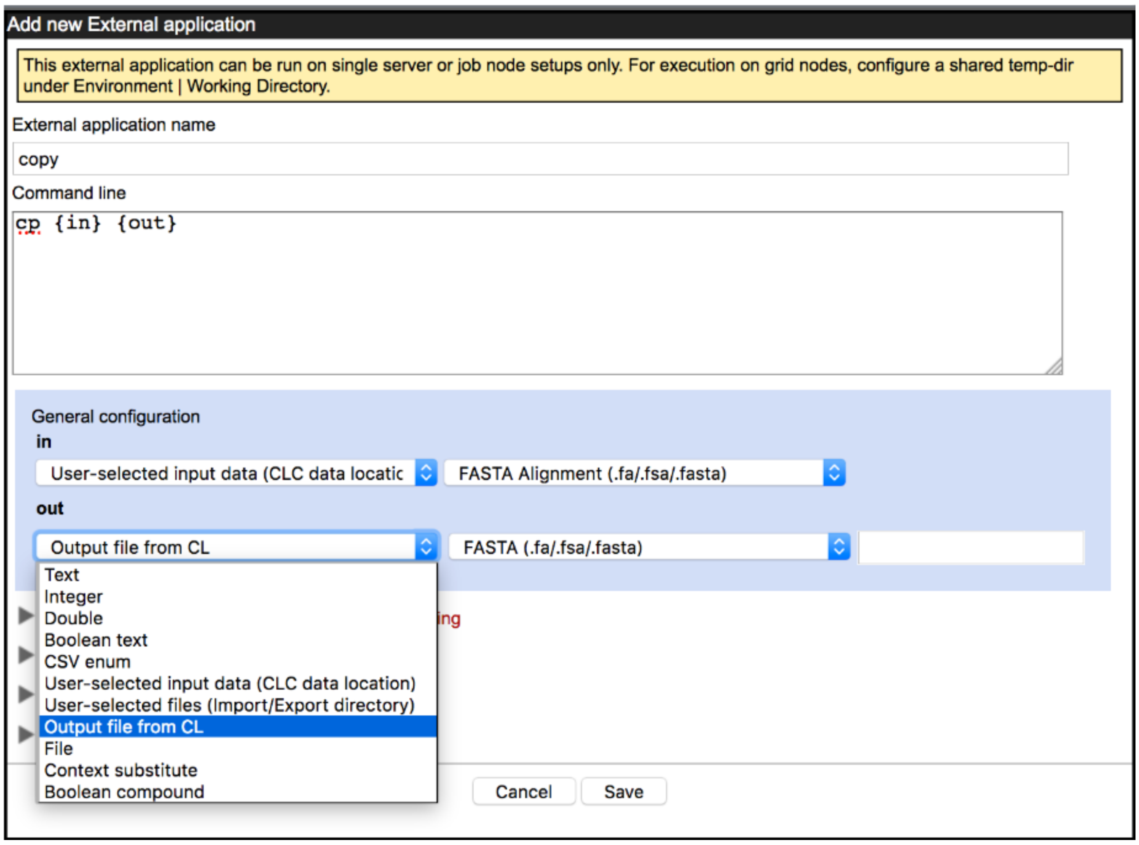
The best way to describe the integration of third party command lines tools is through a series of examples. We start with a very basic example and work towards more complex setups.

12.1 Configuration overview

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

The *cp* command requires at least 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 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 Server* administrative web interface, click on the *New configuration* button. This brings up a window like the one shown in figure 12.2. In the text box labeled *External applications command name*, enter a name for this command. This will be what the end-users see in the menu option that 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. Figure 12.2 shows how this looks if we give the cp command two parameters: *in* and *out*.


Add new External application

This external application can be run on single server or job node setups only. For execution on grid nodes, configure a shared temp-dir under Environment | Working Directory.

External application name: copy

Command line:

```
cp {in} {out}
```

General configuration

in: User-selected input data (CLC data locatic — FASTA Alignment (.fa/.fsa/.fasta)

out: Output file from CL — FASTA (.fa/.fsa/.fasta)

Figure 12.2: *Setting up the cp command as an external application.*

Two drop-down menus have now appeared in the blue shaded area of the window in figure 12.2. 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 are presented with a text box allowing them to enter a parameter value.

The administrator can provide a default value if desired.

- Integer - Allows the user to enter a whole number. The administrator can choose a number this option should be set to by default. If none is set, then 0 is the default.
- 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.
- Boolean text - This generates a checkbox in the end-user interface, labeled with the name of the parameter. If the user checks the box, the parameter is replaced with the given text. If the box is unchecked, the parameter will be empty.
- CSV enum - This allows the administrator to set a drop down list of parameter choices for the user. When selecting the CSV enum type, two text boxes are used to define the set of items that will be presented to the user. The first box defines the actual values in a comma separated list. The second text box defines the labels to display for each of the values. Again the items are comma separated. Make sure the values are unique and that a label is defined for each value.

  For an example of this, please see section section 12.8 on setting up Velvet as an external application.
- User-selected input data (CLC data location) - Users will be prompted to select an input file from those they have stored on the CLC Server.
- User-selected files (Import/Export directory) - Users will be prompted to select one ore more input files among those stored in an Import/Export directory on the CLC Server. This will typically be non-.clc-files. Preselected files can be set.
- 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.
- Context substitute - This parameter is only for the Command Line and is not visible to the end user. The parameter results in substitution by a value in the context where the Command Line is run. Options are: CPU limit max cores: The configured limit at the server executing the Command Line. Name of user: The name of the user that has initiated the External Application.
- 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 grayed out. Here, the administrator can also choose if the box is to be checked or unchecked by default in the Workbench interface.

In the right hand side of figure 12.2 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 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 CLC Workbenches with the *CLC Workbench Client plugin* installed. If a particular external application needs to be removed from end-user access for a while, this small box can just be unchecked.

12.2 Post processing

Once the external application has run and completed successfully, the results can be processed using CLC tools. This could be to carry out an analysis task or to use an importer that needs extra configuration information to get the results back into the *CLC Server*, for example, NGS data importers. Parameters defining outputs of the external application to be handled this way must have the type "Output file from CL". The option "Do not standard import / map to high-throughput sequencing importer" should be chosen from the drop down list of the second field. Each output from the external application that should be handled this way will need a post processing tool configured for it.

A general description of post processing is given in this section. For further details, please refer to section 12.9, which covers configuring Bowtie as an external application. There, two post processing tools are configured for the import of the results.

Post processing tools need to know what data they should use and what values should be used for the parameters. To configure these things, click on the "Edit and map parameters..." button in the High-throughput sequencing import / Post processing section.

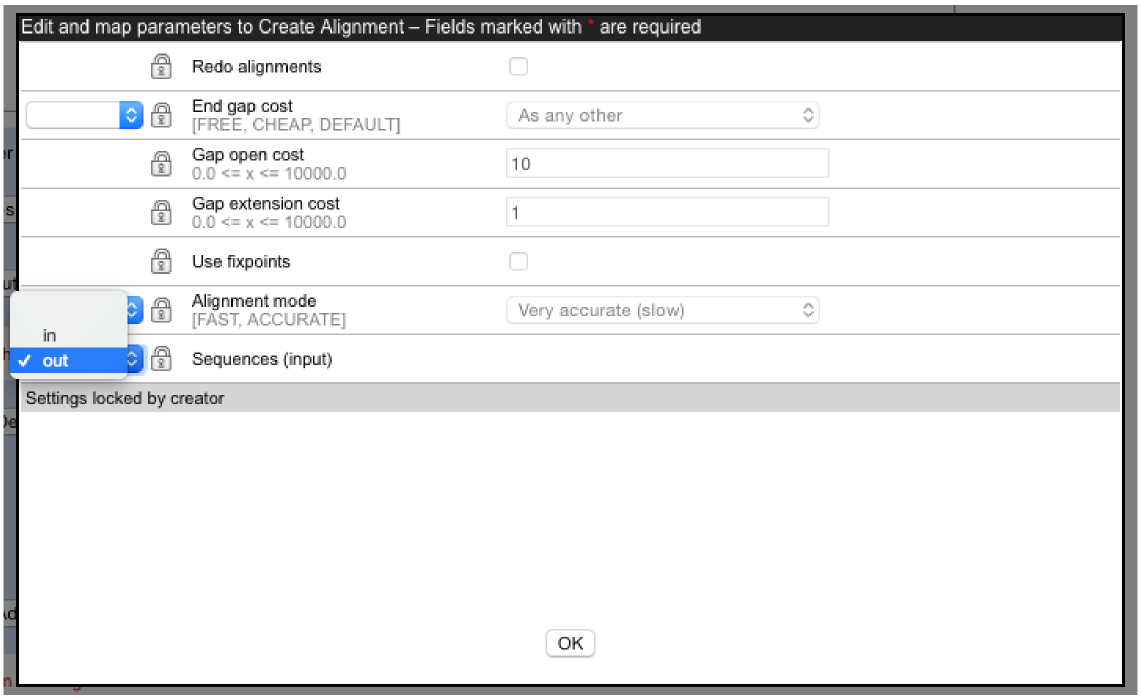
To define the input to a tool, we need to map the relevant output from the external application to the relevant input of the post processing tool. The mapping of one parameter to another is constrained by the type compatibility of the parameters. Those that make technical sense, based purely on the type of the parameter, will be presented in a drop-down list to the left of the relevant post processing tool parameters. See figure 12.3, where parameters called "in" and "out" are available to choose to map as Sequences (inputs) parameter of the Create Alignment tool. Here, the parameter chosen is "out".

After mapping outputs to post processing tools, the text displayed in the second drop down field of the relevant parameter in the General Configuration panel will indicate which post processing tool those outputs are mapped to. For example, if the post processing tool: "Import SAM/BAM Mapping Files" was configured and the relevant output mapped to it, the text in the General Configuration area for the relevant output would show: "Linked with Import SAM/BAM Mapping Files" in the second field.

Whether or not a given parameter for a post processing tool can be configured by end-users is also defined in the "Edit and map parameters..." editor. This is done by clicking on the lock image next to a parameter to either lock it or unlock it.

Before unlocking a parameter, the default value used by the tool is visible but cannot be changed. After unlocking a parameter, you can set a new default value.

If you leave the option unlocked, the end users of the external application will be able to alter that value. If you alter the value and lock the parameter again, the value you provided will be used whenever a user runs the external application and the end-user will not be presented with



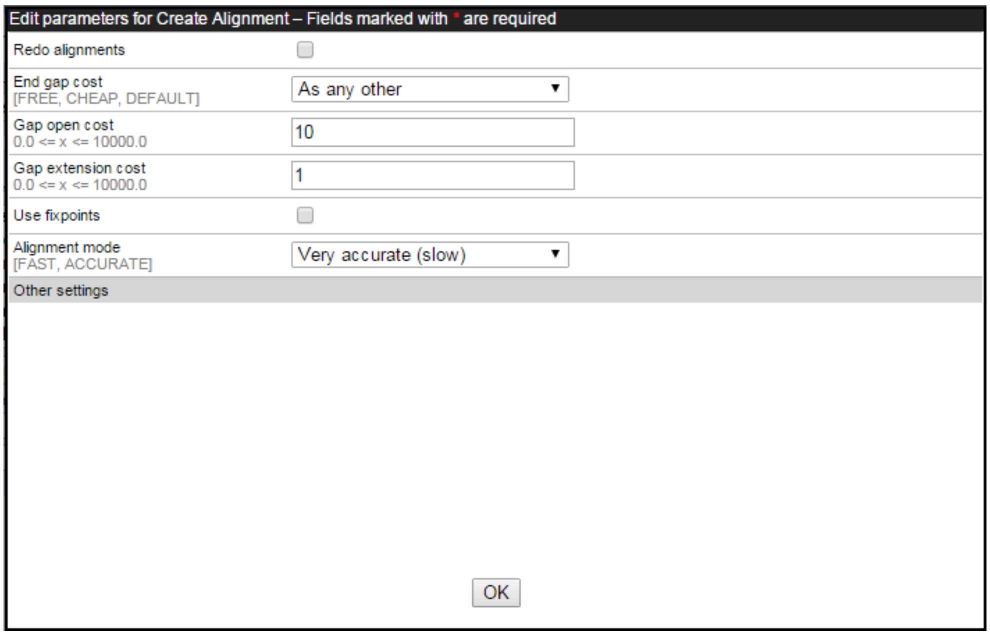
Edit and map parameters to Create Alignment – Fields marked with * are required

Redo alignments	☐
End gap cost [FREE, CHEAP, DEFAULT]	As any other
Gap open cost 0.0 <= x <= 10000.0	10
Gap extension cost 0.0 <= x <= 10000.0	1
Use fixpoints	☐
Alignment mode [FAST, ACCURATE]	Very accurate (slow)
Sequences (input)	in [✓ out]

Settings locked by creator

Figure 12.3: *The output of the external application, here called "out", is being mapped to the input of the post processing tool, the Create Alignment tool.*

it or be able to alter it (figure **??**).



Edit parameters for Create Alignment – Fields marked with * are required

Redo alignments	☐
End gap cost [FREE, CHEAP, DEFAULT]	As any other
Gap open cost 0.0 <= x <= 10000.0	10
Gap extension cost 0.0 <= x <= 10000.0	1
Use fixpoints	☐
Alignment mode [FAST, ACCURATE]	Very accurate (slow)

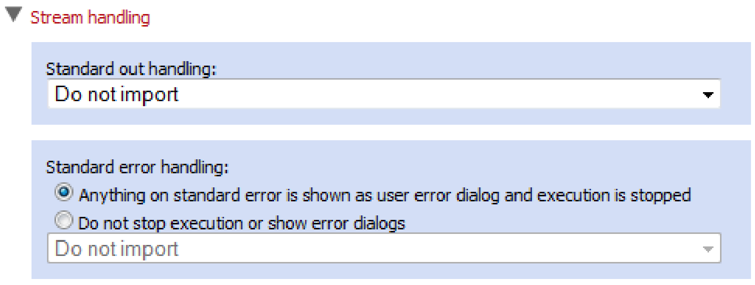
Other settings

Figure 12.4: *The administrator here has unlocked the Gap extension cost parameter, and has left the default value at 1. With this configuration, users will be presented with this parameter, which can be edited when they launch the external application. parameters.*

12.3 Stream handling

There is a general configuration of stream handling available.

The stream handling shown in figure 12.5 allows you to specify where standard out and standard error for the external application should be handled.


Stream handling

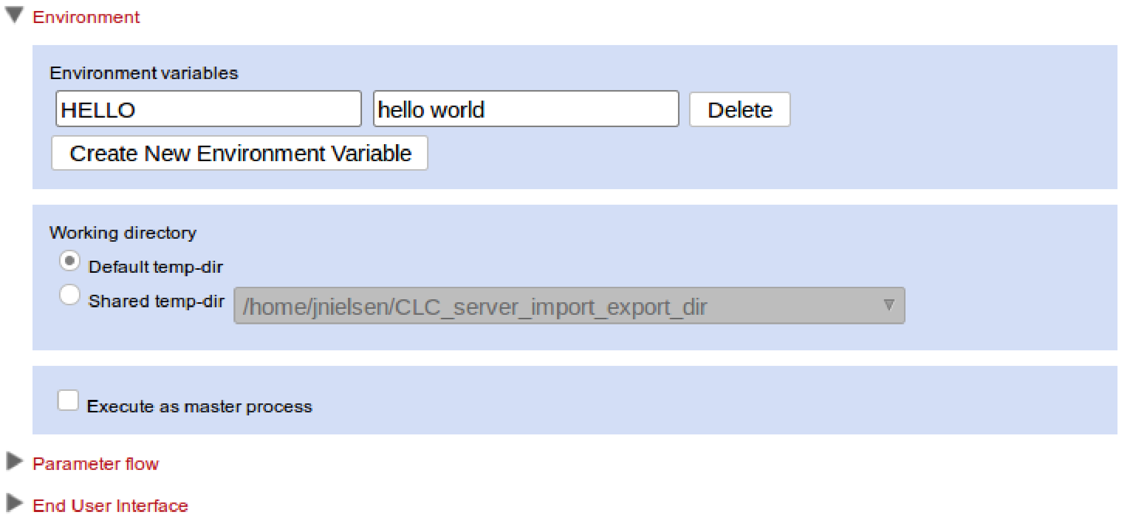
Standard out handling: Do not import

Standard error handling:
(●) Anything on standard error is shown as user error dialog and execution is stopped
( ) Do not stop execution or show error dialogs
Do not import

Figure 12.5: *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.

12.4 Environment


Environment

Environment variables
HELLO | hello world | Delete
Create New Environment Variable

Working directory
(●) Default temp-dir
( ) Shared temp-dir /home/jnielsen/CLC_server_import_export_dir

☐ Execute as master process

Parameter flow
End User Interface

Figure 12.6: *The menu for configuring the execution environment of the external application.*

In the *Environment* sub-menu of the External Applications integration, it is possible to configure a few central aspects of the environment in which the external application will execute. A screenshot of the sub-menu can be seen in Fig. 12.6.

12.4.1 Environmental Variables

In this section it is possible to create and set a default value for environment variables that should be present for the external application when it executes. In the example of Fig. 12.6 an environment variable named "*HELLO*" with value "*hello world*" will be present in the execution environment of the external application.

12.4.2 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.

For external applications that will be run on job nodes, one can choose either the *Shared temp-dir* or the *Default temp-dir* option. Here, the *default temp-dir* would not normally be an area shared between machines, and thus the choice of the *Default temp-dir* means that files will be moved between the master and job node(s).

For external applications that will be run on grid nodes, the *Share temp-dir* option **must** be chosen for the working directory.

If you configure the *Shared temp-dir* for an external application, this area must:

- Be configured in the Import/Export directories area under the Main Configuration tab (see section 3.3).
- Be a shared directory, accessible to your all machines that will execute the external application.

12.4.3 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.

12.5 Running External Applications

External applications can be executed from both a *CLC Workbench* and the *CLC Server Command Line Tools*.

12.5.1 Running from a CLC Workbench

External applications are executed from the CLC Genomics Workbench by going to:

Toolbox | External Applications ( )

Note that external applications are only synchronized between the server and the workbench during login thus users will have to relogin to discover new external applications.

The configured and accessible external applications are available as individual tools as seen in figure 12.7.

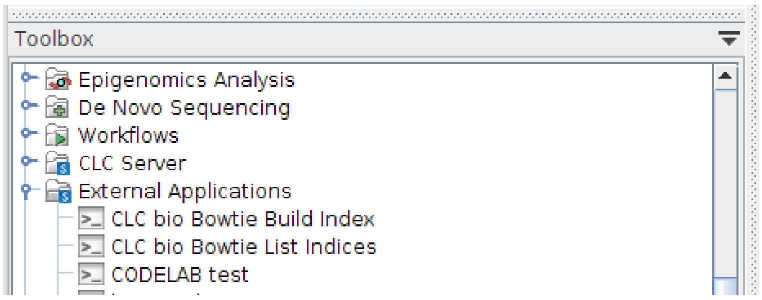


Figure 12.7: *Selecting the external application to run.*

When the *External Applications* item is launched, the dialog shown in figure 12.8 is displayed. There are two types of execution environment: The *CLC Server* environment, and grid presets. The CLC Server environment is always present, while grid presets are only shown if they have been configured as described in section 6.3.

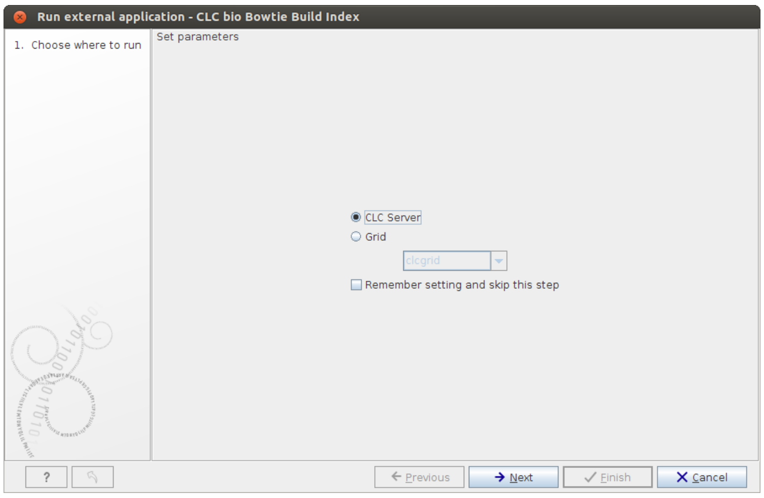

Run external application - CLC bio Bowtie Build Index
1. Choose where to run
Set parameters
CLC Server
Grid
clcgrid
Remember setting and skip this step
?
Previous
Next
Finish
Cancel

Figure 12.8: *Selecting execution environment.*

In order for an external application to be executable in a grid environment, its working directory must be configured as a shared temp-dir (see section 12.4.2 for details).

Clicking *Next* will display the wizard for providing values for the parameters as can be seen in figure 12.9. Here the *in* parameter is displayed as a CLC Object selection, as configured in section 12.1.

12.5.2 Running from CLC Server Command Line Tools

The CLC Server Command Line Tools (CLT) allows algos, workflows and external applications to be invoked on a CLC Server from a command line and scripts. See the *CLC Server Command Line Tools User Manual* for more information.

When using the CLT to run external applications, the server exectution context is chosen unless the `-G` is used to select a specific grid preset. As always, running the CLT with missing or invalid parameters, will provide a help text describing how to correct the situation. Trying to invoke the *copy* external application with no arguments yields the following output:

```
clcserver -S <HOSTNAME> -U <USER> -W <PASSWORD> -A copy
Message: Trying to log on to server
Message: Login successful
```

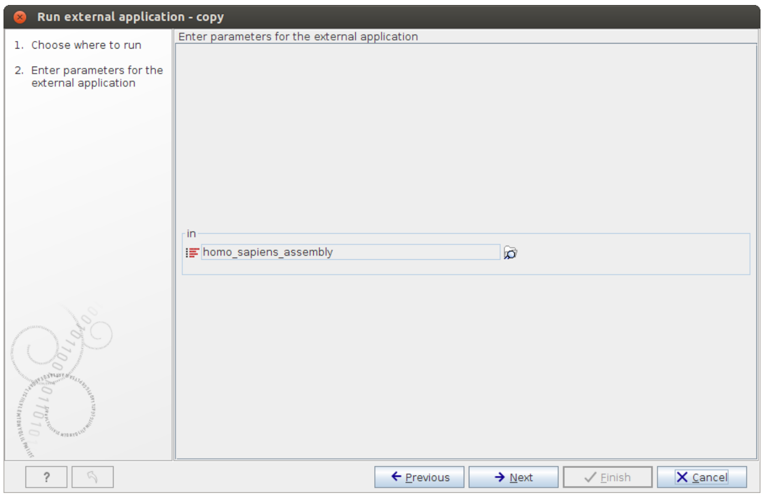

Run external application - copy
1. Choose where to run
2. Enter parameters for the external application
Enter parameters for the external application
in
homo_sapiens_assembly
?
Previous
Next
Finish
Cancel

Figure 12.9: *Providing the parameters to the external application.*

```
The following options are available through the command line and the types are as follows:

Type                                      Valid input
----                                      -----------
<Integer>                                 A decimal number in the range
                                            [-2147483648;2147483647]
                                          Example: 42
<Boolean>                                 The string true or false
                                          Example: true
<String>                                  Any valid string. It is recommended
                                            to enclose all strings in '' to
                                            avoid issues with the shell
                                            misinterpreting spaces or double
                                            quotes
                                          Example: 'text="My text"'
<ClcFileUrl>                              A valid path to a file on the server
                                            or in the local file system
                                          Example: clc://serverfile/tmp/export
<ClcObjectUrl>                            A valid path to a Clc object on the
                                            server or locally
                                          Example: clc://server/pstore1/Variant1

Option                                    Description
------                                    -----------
-A <Command>                              Command currently set to 'copy'
-C <Integer>                              Specify column width of help output.
-D <Boolean>                              Enable debug mode (default: false)
-G <Grid preset names>                    Specify to execute on grid.
-H                                        Display general help.
-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.
-d, --destination <ClcServerObjectUrl>    Destination for import from External
                                            Application
--in <ClcServerObjectUrl>                 Model object(s) to be exported to
                                            FASTA (.fa/.fsa/.fasta)
Error: Missing required options: d, in
```

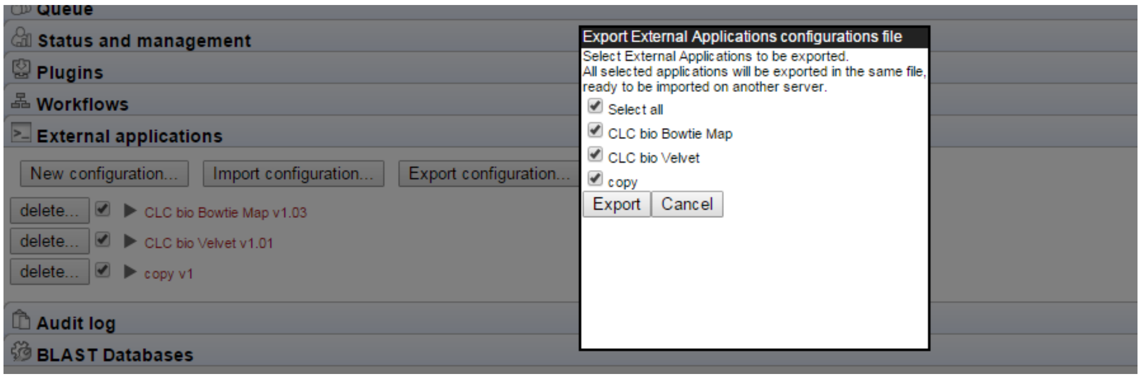
Here we need to give both the `-d` and `-in` parameters in order for the external application being able to run.

12.6 Import and export

External application configurations can be exported and imported in order to facilitate backup and exchange of these configurations between servers. Please note that any installed application used by an external application configuration is not part of the export. Only the execution configuration set up in the administration interface is included.

12.6.1 Export

To export external application configurations, click the **Export configuration. . .** button.



Queue
Status and management
Plugins
Workflows
External applications
New configuration... | Import configuration... | Export configuration...
delete... ✓ ▶ CLC bio Bowtie Map v1.03
delete... ✓ ▶ CLC bio Velvet v1.01
delete... ✓ ▶ copy v1
Audit log
BLAST Databases

Export External Applications configurations file
Select External Applications to be exported.
All selected applications will be exported in the same file, ready to be imported on another server.
✓ Select all
✓ CLC bio Bowtie Map
✓ CLC bio Velvet
✓ copy
Export | Cancel

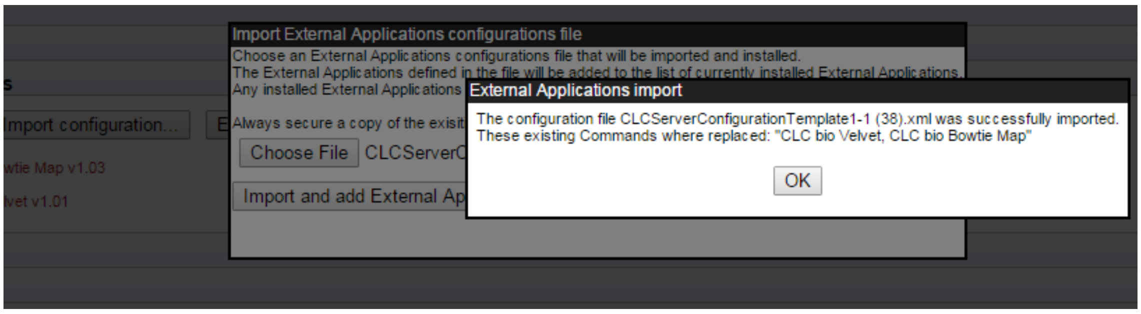
Figure 12.10: *Exporting external applications configuration.*

It is possible to export all external applications at once or to hand pick a subset to be exported. Select the applications to export and click **Export**. A configuration file is then downloaded and can be imported on the same or another server.

12.6.2 Import

To import a set of one or more external applications, click the **Import configuration. . .** button. Select the configuration to import and click **Import**.

You will then see a dialog confirming the import. If any of the imported configurations already existed, they have been overwritten and listed in the dialog.



Import External Applications configurations file
Choose an External Applications configurations file that will be imported and installed.
The External Applications defined in the file will be added to the list of currently installed External Applications.
Any installed External Applications
Always secure a copy of the exisit
Choose File CLCServerC
Import and add External Ap

External Applications import
The configuration file CLCServerConfigurationTemplate1-1 (38).xml was successfully imported.
These existing Commands where replaced: "CLC bio Velvet, CLC bio Bowtie Map"
OK

Figure 12.11: *Exporting external applications configuration.*

12.7 Use of reference data in external applications

The reference data manager can export a reference data set to an import/export location for use in external applications. The manual for *Biomedical Genomics Workbench* explains how to do this using the *Biomedical Genomics Workbench*. However, some configuration needs to be carried out on the *CLC Genomics Server*, in order for this to work seamlessly.

Reference Data Sets are exported to a file system folder structure which, relative to som import/rexport folder, is equivalent to the folder structure below **CLC_References**. This convention makes it possible for an external application to access reference data based on a CLC URL in object name form.

Therefore the server needs to have at least one import/export location set-up, otherwise it is not possible to export reference data. The import/export location can be set-up in the thin client for the CLC Server.

12.7.1 How to make use of reference data in external applications

The external application gets a CLC URL pointing to:

```
CLC_References/<organism>/<data type>/<version>/<filename>
```

and must convert it, using string manipulation, to the following file path:

```
<import/export location>/<organism>/<data type>/<version>/<filename>
```

An example of the above translation could be a CLC object, persisted in the file-system as the file:

```
/opt/gatk/CLC_References/homo_sapiens/sequence/hg19_chr_5/Homo_sapiens_sequence_hg19.clc
```

That CLC object is pointed to by this CLC URL:

```
clc://server/CLC_References/homo_sapiens/sequence/hg19_chr_5/Homo_sapiens_sequence_hg19
```

The external application making use of the exported version of this CLC reference would then have to convert the CLC URL to this file path:

```
/impexp/CLC_References/homo_sapiens/sequence/hg19_chr_5/Homo_sapiens_sequence_hg19.fasta
```

by replacing the `"clc://server"` part of the URL with `"/impexp"`. In this example, it is assumed that `"/impexp"` is the file path of the import/export location, where the reference data has been exported to.

An important note about CLC URLs is that in addition to the more human-readable forms of CLC URL that we have seen above, a CLC URL can equally well take on an ID based form, such as:

```
clc://server//[...]1073273041-BAAAAAAAAAAAAAP90a0346df401e2e8-448c7e40-152545ded0d-8000
```

Obviously, the external application framework would have a hard time translating this URL to anything meaningful in an external context. This challenge has been solved by letting the external application framework always generate an object name form URL (in contrast to an id based URL), when the external application argument has type CLC Object url. Thus, the id based URLs will never be sent to the external application by the external application framework – only the name based form will.

12.7.2 Deletion of exported reference data

Deletion of exported data cannot be done through the workbench, nor the CLC server. It has to be done through the operating system.

12.8 Velvet Integration

Velvet [Zerbino and Birney, 2008] is a popular de novo assembler for next-generation sequencing data.

The velvet package includes two programs that need to be run consecutively. The external applications system on the *CLC Server* is designed to call one program, so a wrapper script is needed to make the needed consecutive calls to the Velvet applications.

An example script as well as a configuration file are available in a zip file at `http://www.resources.qiagenbioinformatics.com/external-applications/velvet-example.zip`. These will be used to illustrate how this sort of application can be configured as an external application on a *CLC Server*.

12.8.1 Installing Velvet

To get started, you need to do the following:

- Install Velvet on the server computer. The program and installation information is available from `https://github.com/dzerbino/velvet/tree/master`. If you have job nodes, Velvet will need to be installed on all the nodes that will be configured to run it.

- Download the scripts and configuration files from `http://www.resources.qiagenbioinformatics` `com/external-applications/velvet-example.zip`. These files have been created assuming that Velvet is installed in `/usr/local/velvet`. If it is installed elsewhere, please update the files with the correct path to the program on your server.

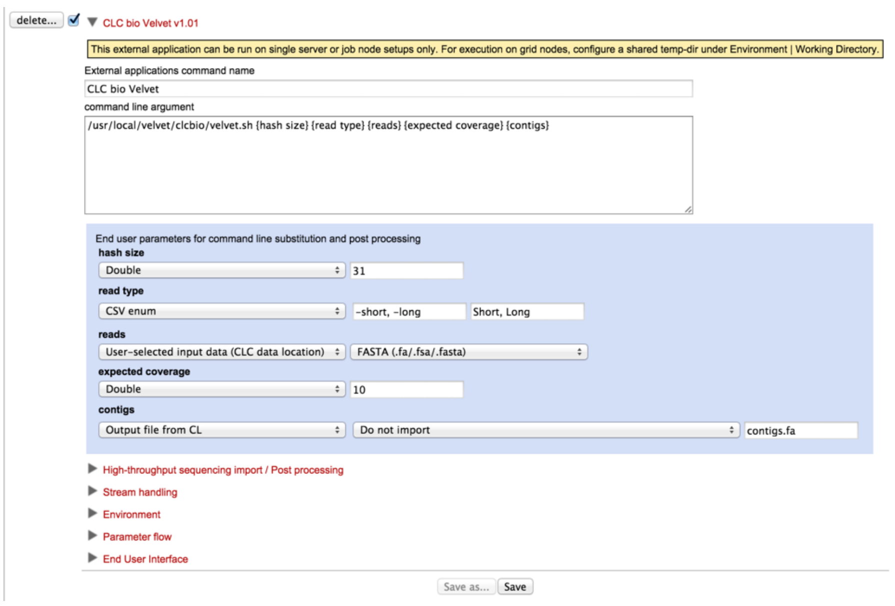
- Check to ensure execute permissions are set on the velvetg and velveth executable files in the Velvet installation directory. These must be executable by the user that owns the CLC Server process.

- Unzip the velvet-example.zip file and place the `clcbio` folder and its contents in the Velvet installation directory. This contains a script (velvet.sh) that links the two Velvet programs, velvetg and velveth, together. If the Velvet binaries are not in the folder `/usr/local/velvet`, you will need to edit the line that starts with exe= to include the correct path.

- Set the permissions on the velvet.sh script in the clcbio subfolder so that it can be executed by the user that owns the CLC Server process.

- 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 12.12.


CLC bio Velvet v1.01

This external application can be run on single server or job node setups only. For execution on grid nodes, configure a shared temp-dir under Environment | Working Directory.

External applications command name: CLC bio Velvet

command line argument:

```
/usr/local/velvet/clcbio/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, -long	Short, Long
reads	User-selected input data (CLC data location)	FASTA (.fa/.fsa/.fasta)	
expected coverage	Double	10	
contigs	Output file from CL	Do not import	contigs.fa

Figure 12.12: *The Velvet configuration has been imported.*

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

If you wish to execute this job on grid nodes, then a shared temp-dir must be specified in the Working directory section of the Execution area of the configuration. See section 12.4.2 for details.

If you see a small red exclamation point beside the external application name, then something is wrong and needs to be attended to. The specific area where there is a problem should also be identified by a red exclamation mark.

This is seen, for example, when the versions of a High-throughput sequencing import or Post-processing step specified in the configuration file is different to the version on the *CLC Server*. This can occur when the configuration was set up on an older version of the *CLC Server* than the one running. This situation is shown in figure 12.13. It is easily resolved by expanding the High-throughput sequencing import / Post-processing section, selecting the relevant tool and then saving the configuration.

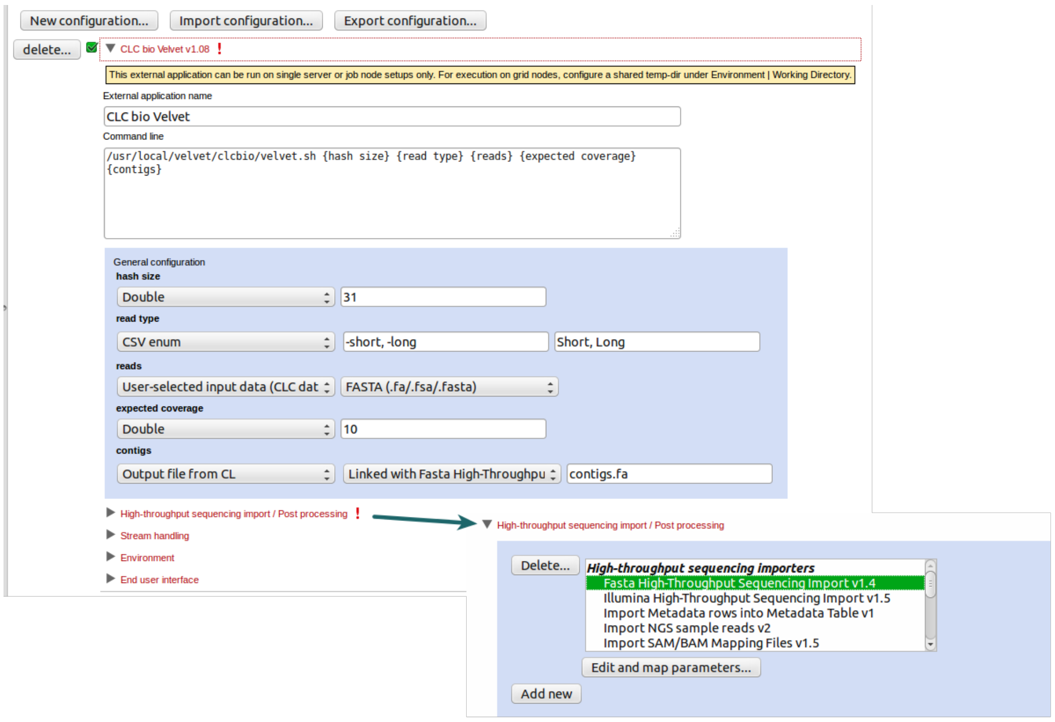
12.8.2 Running Velvet from the Workbench

To run Velvet, open a Workbench with the **External Applications Client Plugin** installed. Then:

- Go to:

  Toolbox | External Applications ( ) | CLC bio Velvet ( )

- Confirm where you wish to run the job.
- Select ( ) the reads to assemble and configure the Velvet parameters, as shown in figure 12.14.
- Click on the button labeled **Next**.
- Specify where to save the results.



New configuration... Import configuration... Export configuration...

delete... CLC bio Velvet v1.08 !

This external application can be run on single server or job node setups only. For execution on grid nodes, configure a shared temp-dir under Environment | Working Directory.

External application name

CLC bio Velvet

Command line

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

General configuration

hash size	Double	31	
read type	CSV enum	-short, -long	Short, Long
reads	User-selected input data (CLC dat	FASTA (.fa/.fsa/.fasta)	
expected coverage	Double	10	
contigs	Output file from CL	Linked with Fasta High-Throughpu	contigs.fa

High-throughput sequencing import / Post processing !

Stream handling

Environment

End user interface

High-throughput sequencing import / Post processing

Delete...

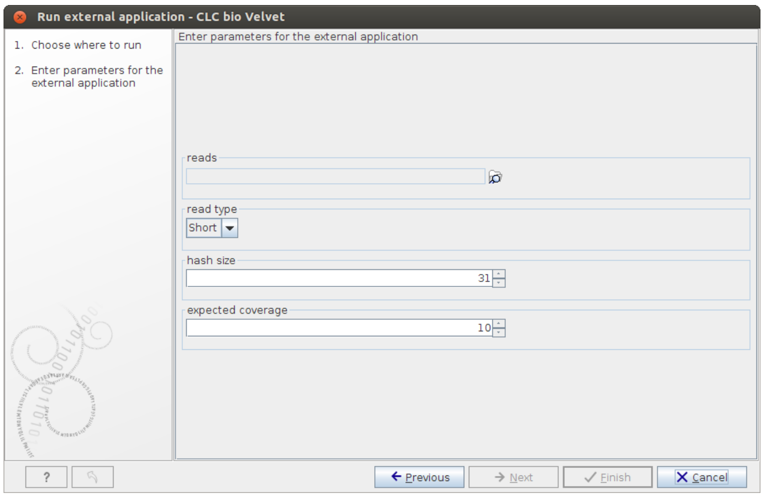
High-throughput sequencing importers

Fasta High-Throughput Sequencing Import v1.4
Illumina High-Throughput Sequencing Import v1.5
Import Metadata rows into Metadata Table v1
Import NGS sample reads v2
Import SAM/BAM Mapping Files v1.5

Edit and map parameters...

Add new

Figure 12.13: *Problems with the configuration are indicated by red exclamation marks, as can be seen here by the external application name and by the area with the problem, the High-throughput sequencing import/ Post-processing section. Expanding this section, the problem can be resolved by selecting the relevant tool.*



Run external application - CLC bio Velvet

1. Choose where to run
2. Enter parameters for the external application

Enter parameters for the external application

reads

read type: Short

hash size: 31

expected coverage: 10

Figure 12.14: *Configuring Velvet parameters from a Workbench.*

- Click on the button labeled **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

12.8.3 Understanding the Velvet configuration

The Velvet configuration file is explained here as a specific example of all external application configuration files.

Going back to figure 12.12, 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 part is the path to the script. The following parts are parameters that are interpreted by the server when calling the script. Parameters to be interpreted 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 parameter, `hash size`, can be entered as a **Double** (which is a number that can take decimal values in computer parlance). The user provides a value when they launch Velvet. A default value is provided in the configuration (31).

The second parameter is `read type`, which has been configured as a **CSV enum** which means a list of possible values that the user can choose from. The first part of the configuration information consists of the parameters to be used when calling the velvet script (`-short, -shortPaired, -long, -longPaired`), and the second part is the more human-readable representation that is to be shown in the Workbench (`Short, Short Paired, Long, Long Paired`).

The third parameter is `reads` which is used for 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 from clc format to a temporary fasta file before running the script.

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

The last parameter, `contigs`, represents the output file, as indicated by the choice of "Output file from CL" as the type. Here, you specify how the output from velvet should be handled. A list of standard import formats is provided, as well as the option not to import using those tools. Choosing not to import using those tools means that you can choose a high throughput importer instead from the section High-throughput sequencing import/ Post-processing section.

For this example, Do not import is the action set for the contigs parameter. Then, below, in the High-throughput sequencing import/ Post-processing section, the Fasta High-throughput Sequencing Import tool has been selected. Thus, when the results from velvet are ready, they are imported into the CLC Server using that tool and saved where the user indicates when they run the job.

12.9 Bowtie Integration

In this example, we show how to integrate Bowtie [Langmead et al., 2009], a popular tool for mapping short sequencing reads to a reference sequence, using the External Applications functionality. Here, two post processing tools will be configured, allowing us to import more than one output generated by bowtie into the CLC Server.

Importing the configuration file provided as an example, and following the instructions in this section, leads to three tools being made available to users logged into the CLC Server via their Workbench or the Command Line Tools: CLC bio Bowtie Build Index, CLC bio Bowtie List Indices and CLC bio Bowtie Map.

12.9.1 Installing Bowtie

To get started:

- Install Bowtie from `http://bowtie-bio.sourceforge.net/index.shtml`. We assume that Bowtie is installed in `/usr/local/bowtie` but you can just update the paths if it is placed elsewhere.
- Download the scripts and configuration files from `http://www.resources.qiagenbioinformatics` `com/external-applications/bowtie-pp2.zip`
- Place the `clcbio` folder and contents in the Bowtie installation directory. This folder contains the scripts used to wrap the Bowtie functionality. Those wrapper scripts are what is then configured via the External Applications folder.
- Make sure execute permissions are set on these wrapper scripts and on the executable files located in the Bowtie installation directory. The user that will execute these files will be the user that started the CLC Server process.
- Import the `bowtie-pp2.xml` file as a new configuration on the server by going to the **External applications** () tab under **Admin** () in the web administrative interface and clicking on the button labeled **Import Configuration**.

The `bowtie-pp2.xml` file contains configurations for three tools associated with Bowtie: `CLC Bowtie build index`, `CLC Bowtie list indices` and `Bowtie Map`. If you already have a set of indices you wish to use and the location of these is known to the system via the BOWTIE_INDEXES, then you can just use the `Bowtie Map` tool via the Workbench and specify the index to use by name.

Otherwise, you can build the index to use using the `CLC Bowtie build index` tool. Here, unless you edit the wrapper scripts in the files you download from CLC bio, the indices will be written to the directory indicated by the BOWTIE_INDEXES environmental variable. If you have not specified anything for this, indices will likely be written into the folder called `indexes` in the installation area of Bowtie. Please ensure that your users have appropriate write access to the area indices should be written to.

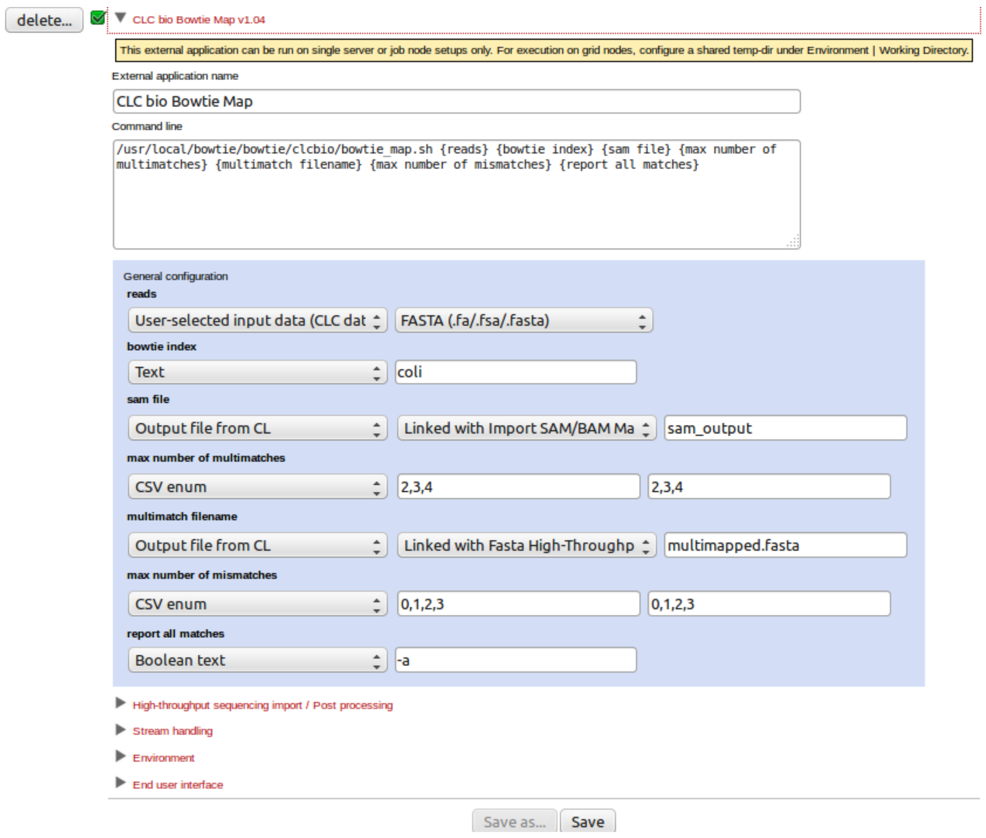
From `ftp://ftp.cbcb.umd.edu/pub/data/bowtie_indexes/` you can download pre-built index files of many model organisms. Download the index files relevant for you and extract them into the `indexes` folder in the Bowtie installation directory.

When configuring Bowtie to run as an external application on master-node setups, the **Environment** configuration will need to be edited. See 12.9.3 for details. In addition, Bowtie indices will have to be placed somewhere accessible to all nodes. One option could be areas configured as Import/Export areas on the *CLC Server*.

The rest of this section focuses on understanding the integration of the `Bowtie Map` tool in particular.

12.9.2 Understanding the Bowtie configuration

Once the `bowtie-pp2.xml` configuration file has been imported, you can click the **CLC bio Bowtie Map** header to see the configuration as shown figure 12.15.



This external application can be run on single server or job node setups only. For execution on grid nodes, configure a shared temp-dir under Environment | Working Directory.

External application name: CLC bio Bowtie Map

Command line:

```
/usr/local/bowtie/bowtie/clcbio/bowtie_map.sh {reads} {bowtie index} {sam file} {max number of
multimatches} {multimatch filename} {max number of mismatches} {report all matches}
```

General configuration

reads		
User-selected input data (CLC dat	FASTA (.fa/.fsa/.fasta)	
bowtie index		
Text	coli	
sam file		
Output file from CL	Linked with Import SAM/BAM Ma	sam_output
max number of multimatches		
CSV enum	2,3,4	2,3,4
multimatch filename		
Output file from CL	Linked with Fasta High-Throughp	multimapped.fasta
max number of mismatches		
CSV enum	0,1,2,3	0,1,2,3
report all matches		
Boolean text	-a	

Figure 12.15: *The Bowtie configuration has been imported.*

From an end-user perspective, when the configuration on the CLC Server is complete, they will be able to launch the CLC bio Bowtie Map tool via their Workbench Toolbox. A wizard will appear, within which they will select the sequencing reads to be mapped, identify the pre-built index file of the reference sequence to use and set a few parameters. The bowtie executable will then be executed on the server system and the results generated will be imported into the CLC Server using post processing tools. The sam mapping file is imported using the Import SAM/BAM Mapping Files tool. A fasta file of sequences mapping to multiple locations is imported using the Fasta High-Throughput Sequencing Import tool.

Below, we step through the General configuration panel and then explain the configuration of the post processing tools that handle the outputs from the bowtie analysis.

General configuration panel

Each of the parameters (items within curly brackets) written into the "Command line" box is presented as an item in the General configuration panel. There, we define the type of information each parameter expects or represents and default values, where relevant.

To understand how these parameters relate to the information that will be passed to the native bowtie executable, please refer to the bowtie_map.sh script in the clcbio folder that should now be in place in the bowtie distribution folder.

Stepping through the parameters in the order they appear in the Command line area of the configuration, and thus the order they appear in the General config panel:

- The `reads` parameter refers to the data that will be provided to bowtie to map. The **User-selected input data** option means the user will be able to select data in a CLC File or Data Location. This data will be exported from the CLC File or Data Location such that the bowtie tool can use it. The second element in this line specifies the format the data should be exported in. This is set to **FASTA (.fa/.fsa/.fasta)** as this is the format the bowtie tool expects sequencing read data.

- The `index` parameter is expecting the name of a bowtie index. Specifying the type **Text** for this parameter means a user will see a box in the Workbench Wizard that they can type text into. Here, a default name, "coli" has been specified, which can be changed by a user launching CLC bio Bowtie Map.

  When setting up a tool like this, it would be simpler for users, and much less subject to error, if the type **CSV enum** were selected, and a specified set of indices were listed. Then, a drop down list of options would be provided to the user in a Workbench Wizard, when launching the external application, rather than relying on users typing in the correct names of available bowtie indices.

- The `sam file` parameter refers to the sam mapping file that bowtie will generate as one of its results file. Thus, the type is set to **Output file from CL**. Import of sam files into the CLC Server involves a tool that requires user input. Thus, a post processing tool is configured. This can be seen immediately by the text in the second drop-down box: "Linked with Import SAM/BAM Mapping Files".

  If a parameter with type **Output file from CL** is not mapped to a parameter of a post processing step, the text displayed is "Do not standard import / map to high-throughput sequencing importer". Mapping of outputs to post processing tools is described in more detail below.

  The last entry in the configuration of the `sam file` parameter is the name of the sam output file that bowtie should generate. Here it is set to **sam_output**. This file name is used by the bowtie command. The Workbench or Command Line Tools user never sees it.

- The `max number of multimatches` parameter allows a user to select the maximum number of locations a read can map equally well to for it to be included in the mapping. The type is set to **CSV enum**, which means a user will be able to select a value from a drop down list of the 3 values listed in the last field (2,3,4). The first value will be the default. The values in the middle field are those passed to the bowtie wrapper script and then onto bowtie. So, for example, if "2,3,4" were entered in the middle field, and "two, three, four" in the last field, a user could select the option "two", and bowtie would be sent the value 2.

- The `multimatch filename` parameter refers to another output from bowtie, this one containing fasta formatted reads that match to multiple locations of the reference equally well. Since it is a result file, the type is set to **Output file from CL**. We have decided to use a post processing tool to bring the results back into the CLC Server, the Fasta High-Throughput Sequencing Import tool.
- The `max number of mismatches` parameter allows a user to select the maximum number of mismatches to be allowed between a read and the reference in order for a read to be considered as matching the reference at that location. The type is set to **CSV enum**, and is presented to a user in the same way as the `max number of multimatches` parameter described above.
- The `report all matches` option is one that can be turned on or off. Thus it is set to type **Boolean text**. A user will be presented with a checkbox they can select or deselect in the Workbench Wizard. The value in the text field, here "-a", is the one bowtie will be passed if the user selects the checkbox. If the user does not select the checkbox, this parameter will not be sent to bowtie.

Post processing - importing the results from bowtie

If you expand click on the **High-throughput sequencing import /Post-processing** link below the General configuration area, you will see that there are two post-procesing tools selected: the **Import SAM/BAM Mapping Files** tool and the **Fasta High-Throughput Sequencing Import** tool.

In each case, clicking on the **Edit and map paramaters** button below it will bring up the configuration window for that tool. Here, several types of configuration can be carried out.

1. Mapping of outputs of the external application to inputs of the post processing tool.
2. Locking or unlocking of parameters, determining which parameters users can alter when launching the tool via the Workbench or Command Line Tools.
3. Setting default values for parameters of the external application.

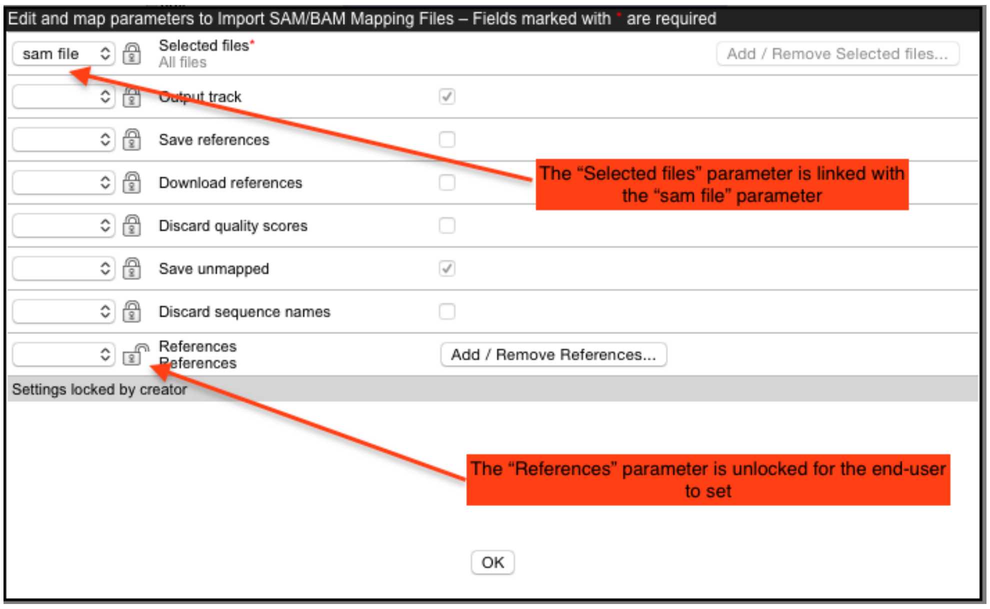
Here, we step through the configuration of the **Import SAM/BAM Mapping Files** tool. The configuration of the **Fasta High-Throughput Sequencing Import** is similar.

The parameters in this configuration window are the **Import SAM/BAM Mapping Files** tool parameters, just as would be offered when that tool is launched directly in a CLC Workbench.

A locked lock symbol by a parameter means that the user will not be given access to this option when launching the tool. Default settings for lock parameters are used. The locked parameters shown in figure 12.16 indicate that a track will be output rather than a stand-alone read mapping, unmapped reads will be saved, references will not be downloaded from an external source and, had they been, downloaded references will not be saved. Quality scores and sequence names will be kept (not discarded).

By contrast, the References parameter is unlocked. When using the Import SAM/BAM Mapping Files tool, users need to specify where the relevant reference sequences are. Thus, this option should be made available for users to configure when the tool is being launched.

The input to the Import SAM/BAM Mapping Files also needs to be defined. This is done by mapping the relevant output from the bowtie command to the input parameter for the Import SAM/BAM Mapping Files tool. The output from bowtie is defined by the "sam file" parameter, and the relevant input parameter in the import tool is "Selected files". A drop down list of potentially relevant parameters appears to the left of the "Selected files" parameter. In our example, this has already been mapped to the "sam file" parameter of the command, as shown in figure 12.16.


Edit and map parameters to Import SAM/BAM Mapping Files – Fields marked with * are required

sam file — Selected files* (All files)	Add / Remove Selected files...
Output track	✓
Save references	☐
Download references	☐
Discard quality scores	☐
Save unmapped	✓
Discard sequence names	☐
References (References)	Add / Remove References...

Settings locked by creator

The "Selected files" parameter is linked with the "sam file" parameter

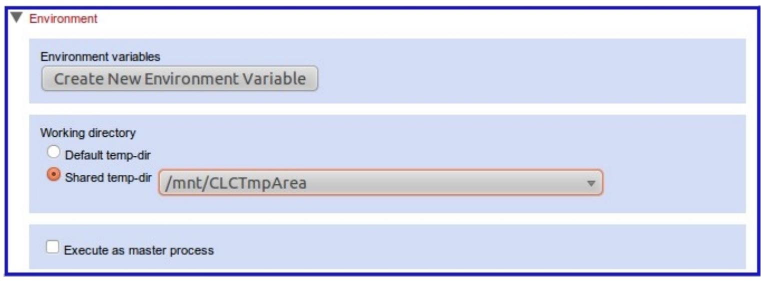
The "References" parameter is unlocked for the end-user to set

Figure 12.16: *Configuration of Import SAM/BAM Mapping Files for import of a sam file after mapping using bowtie.*

Note: The drop-down lists of possibly relevant parameters provided in the post processing tool configuration window are populated based only on the types of parameters (in the General config pane). Any parameters of a type that could be relevant are presented. This means that some parameters appearing in these lists may not make sense contextually.

12.9.3 Setting path for temporary data

The **Environment** handling shown in figure 12.17 allows you to specify a folder for temporary data and add additional environment variables to be set when running the external application.



Working directory
Default temp-dir [✓ Shared temp-dir] /mnt/CLCTmpArea

☐ Execute as master process

Figure 12.17: *Where to save temporary data needs to be defined for Bowtie.*

Post-processing steps need to access the results files of the external application. Thus, **if you are running on a master-node setup, the directory you choose for these results files must be shared**, that is, accessible to all nodes you plan to have as execution nodes for this external application. This is because different stages of your task could be run on different nodes. For example, the export process could run on a different node than the actual execution of the Bowtie script and the post processing. Thus, in a master-node setup, be it using grid or CLC execution nodes, having this shared temporary area eliminates the overhead of transferring the temporary files between nodes.

12.9.4 Tools for building index files

We have also included scripts and configurations for building index files using the external applications on *CLC Server*. This also includes the possibility of listing the index files available. To get these to work, please make sure the path to the Bowtie installation directory is correct.

The Bowtie distribution itself also includes scripts to download index files of various organisms.

12.10 External Applications in Workflows

Like most other tools available for execution on the *CLC Server*, tools configured as external applications can be included in Workflows. Parameters that are locked in a Workflow element will not be offered to the user for editing. Parameters that are unlocked will be.

The inputs and outputs are as configured in the external application itself. In figure 12.18, the external application, CLC bio Bowtie Map is used as a Workflow element. In this particular case, the configuration had a single post processing step, the Import SAM/BAM Mapping Files tool. That tool can output a stand alone read mapping or a read mapping track, and outputs a sequence list containing unmapped reads. Thus these are the output options you see in the Workflow element.

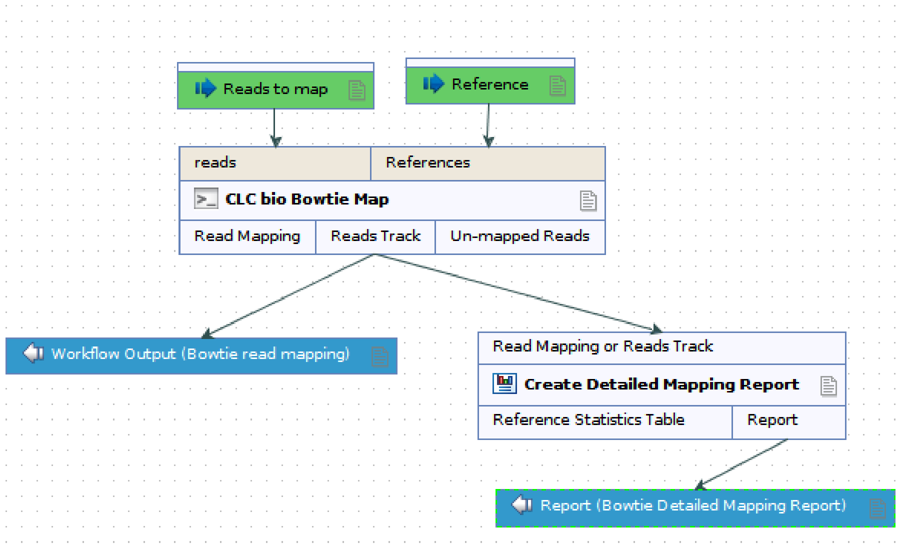

Reads to map
Reference
reads
References
CLC bio Bowtie Map
Read Mapping
Reads Track
Un-mapped Reads
Workflow Output (Bowtie read mapping)
Read Mapping or Reads Track
Create Detailed Mapping Report
Reference Statistics Table
Report
Report (Bowtie Detailed Mapping Report)

Figure 12.18: *The CLC bio Bowtie Map external application used as an element in a Workflow. Here, the outputs visible are those provided by the (single) post processing tool configured: Import SAM/BAM Mapping Files*

When additional post processing tools are configured, their outputs will be added to those available in the Workflow element. See figure 12.19, where a Workflow element for the external application, CLC bio Bowtie Map is present, but in this case, the configuration specified two post processing tools: the Import SAM/BAM Mapping Files tool and the Fasta High-Throughput Sequencing Import tool. As earlier, the three outputs associated with the Import SAM/BAM Mapping Files tool are present. In addition, the output from the Fasta High-Throughput Sequencing Import tool, "Imported reads" is present.

The output channel names make sense for individual tools, but may not make sense in the context of a given external application. For example "Imported Reads" makes sense when you run the Fasta High-Throughput Sequencing Import tool by itself. However, in the context of the CLC bio Bowtie Map external application, it is not indicative of what is really being output. Meaningful names can be set on the Workflow output elements themselves though, providing more information to the Workflow user about what a particular output is. See for example figure 12.19, where the output from the Fasta High-Throughput Sequencing Import tool, "Imported reads", has been renamed "bowtie multimapped reads".

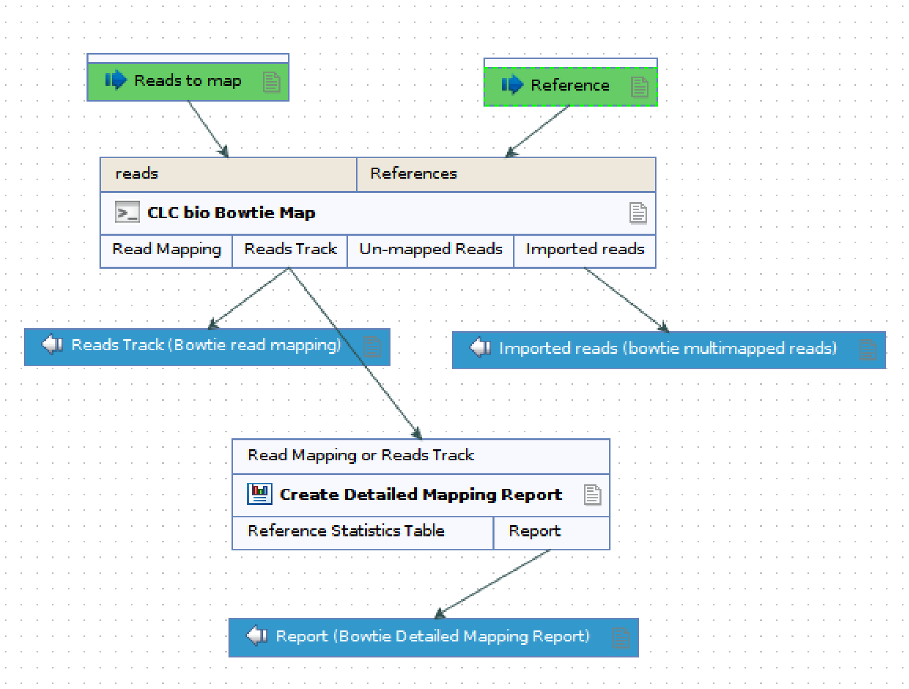

Reads to map
Reference
reads
References
CLC bio Bowtie Map
Read Mapping
Reads Track
Un-mapped Reads
Imported reads
Reads Track (Bowtie read mapping)
Imported reads (bowtie multimapped reads)
Read Mapping or Reads Track
Create Detailed Mapping Report
Reference Statistics Table
Report
Report (Bowtie Detailed Mapping Report)

Figure 12.19: *The CLC bio Bowtie Map external application used as an element in a Workflow. Here, the outputs visible are those provided by the post processing tools configured: Import SAM/BAM Mapping Files and Fasta High-Throughput Sequencing Import.*

Limitation: External applications can only be used in a workflow if they are configured with at least one "User-selected input data" and at least one import of the data generated. If the external application does not generate data that is suitable for import then importing the output from standard out or standard error as plain text is recommended. In general it is recommended to always import standard error to help identify potential problems with an external application.

Please refer to a CLC Workbench manual http://www.qiagenbioinformatics.com/support/manuals/ for more details about designing and running Workflows.

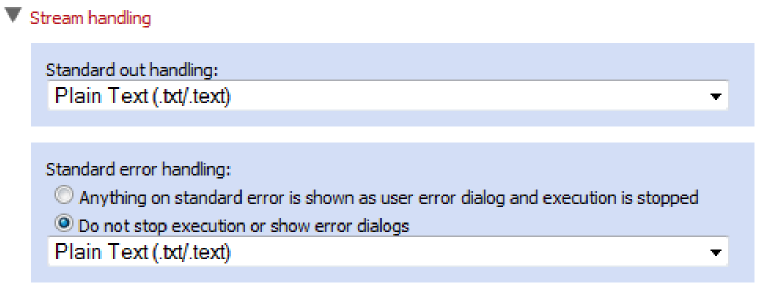
12.11 Troubleshooting external applications

12.11.1 Checking the configuration

There is no check for the consistency of the configuration when it has been set up, so errors will only be seen on runtime when the application is executed. In order to help with troubleshooting, there are a few things that can be done:

Users of an external application launched via a CLC Workbench will see an error window if something goes wrong. Information in the Message tab may help to identify the issue. If not, try opening the **Advanced** tab, where the error message from the system should be visible.

Another aid when debugging is 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 12.20).


Stream handling

Standard out handling:
Plain Text (.txt/.text)

Standard error handling:
( ) Anything on standard error is shown as user error dialog and execution is stopped
(●) Do not stop execution or show error dialogs
Plain Text (.txt/.text)

Figure 12.20: *Importing the result of standard error and standard out.*

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

If your external application was previously working and then stops working, you can also see if the configuration has been recently changed, for example, by another administrator. Clues that this may have happened include

- The version number of the external application has changed. The version number is bumped each time the configuration is saved.
- When mousing over the name of the external application in the thin client, the timestamp of the most recent change to the configuration is presented beside the name of the user who made the change.

A "Change server configuration" operation is recorded in the audit log when changes to external application configurations are made. Clicking on the links in the log entry in the Data In column, you can see details related to the configuration change made.

12.11.2 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?

12.11.3 Is your Import/Export directory configured?

For certain setups, you need to have Import/Export directories configured. Please refer to section 12.4.2 for more details on this.

12.11.4 Check for conflicts in the naming of the external application

If your users will only access 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 Server*, then please ensure that you do not use the same name as any of the *CLC Server* internal commands. You can get a list of these by running the clcserver command, with your *CLC Server* details, and using the flag with no argument. I.e. a command of the form:

clcserver -S <host> -P <port> -U <username> -W <password or token>

Chapter 13

Workflows

The *CLC 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. For a workflow to be executable on a *CLC Server*, all tools in the workflow should be available on the server.

The workflow is created in the CLC Workbench and an installer file is created that can be installed on the *CLC Server*.

As an example from *CLC Genomics Workbench* or *Biomedical Genomics Workbench*, 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.

13.1 Installing and configuring workflows

Workflows can be installed from the server web interface:

> **Admin** ( ) | **Workflows** ( )

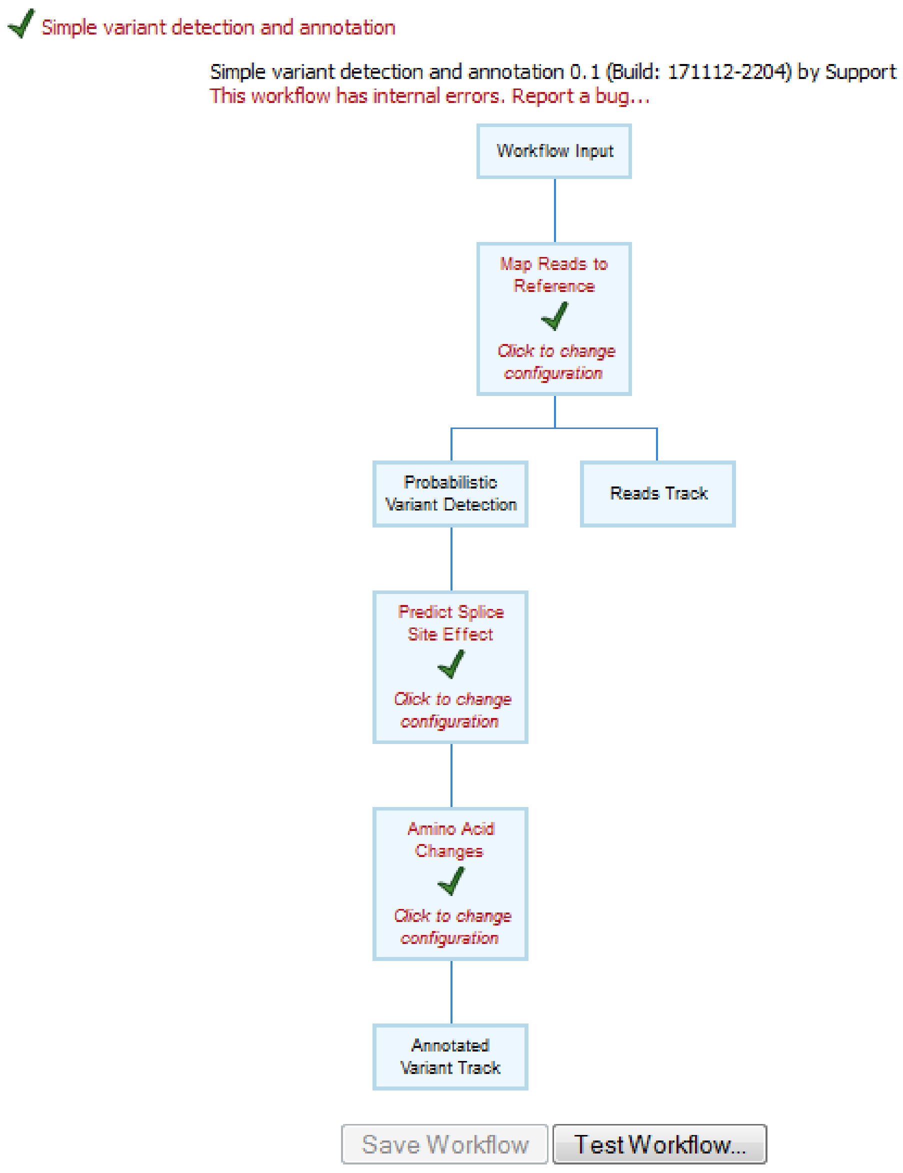
Click the **Install Workflow** button and select a workflow installer (for information about creating a workflow, please see the user manual of *CLC Genomics Workbench*, *Biomedical Genomics Workbench*, *CLC Main Workbench* or *CLC Drug Discovery Workbench* at http://www.qiagenbioinformatics.com/support/manuals/).

Once installed, the workflow is listed with a validated (✓) or attention (❗) status icon as shown in figure 13.1.

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 13.2.

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 Workbench user manuals, which can be found at http://www.qiagenbioinformatics.com/support/manuals/.

If changes to the parameter settings of an installed workflow are made, the timestamp of the


Simple variant detection and annotation

Simple variant detection and annotation 0.1 (Build: 171112-2204) by Support
This workflow has internal errors. Report a bug...

Workflow Input
Map Reads to Reference
Click to change configuration
Probabilistic Variant Detection
Reads Track
Predict Splice Site Effect
Click to change configuration
Amino Acid Changes
Click to change configuration
Annotated Variant Track
Save Workflow
Test Workflow...

Figure 13.1: *A workflow is installed and validated.*

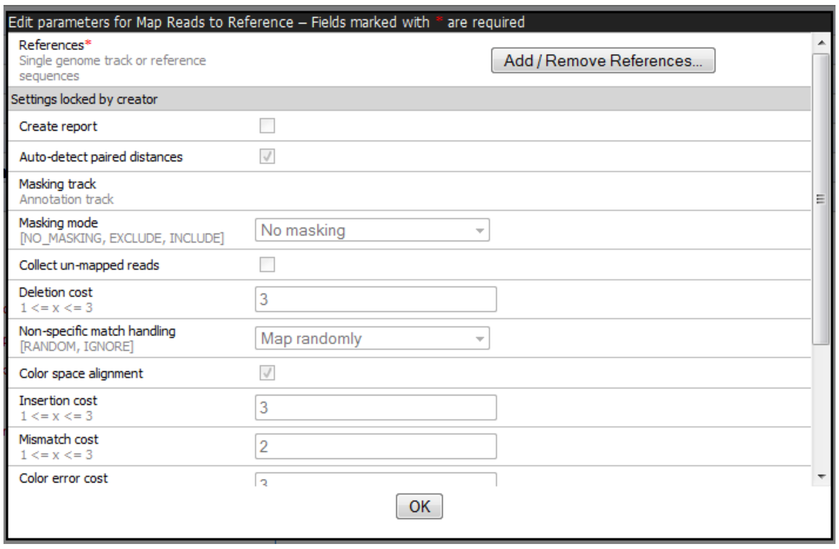
most recent change and the name of the administrator who made those changes are reported at the top of the workflow configuration view.

13.2 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 available in the **Toolbox** (see figure 13.3).

When you select it, you will be presented with a dialog as shown in figure 13.4 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 in the workbench can be executed on the



Edit parameters for Map Reads to Reference – Fields marked with * are required

References*
Single genome track or reference sequences
Add / Remove References...

Settings locked by creator

Create report	☐
Auto-detect paired distances	✓
Masking track Annotation track	
Masking mode [NO_MASKING, EXCLUDE, INCLUDE]	No masking
Collect un-mapped reads	☐
Deletion cost 1 <= x <= 3	3
Non-specific match handling [RANDOM, IGNORE]	Map randomly
Color space alignment	✓
Insertion cost 1 <= x <= 3	3
Mismatch cost 1 <= x <= 3	2
Color error cost	3

OK

Figure 13.2: *In this example only one parameter can be configured, the rest of the parameters are locked for the user.*

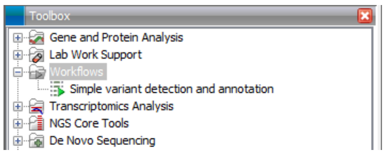
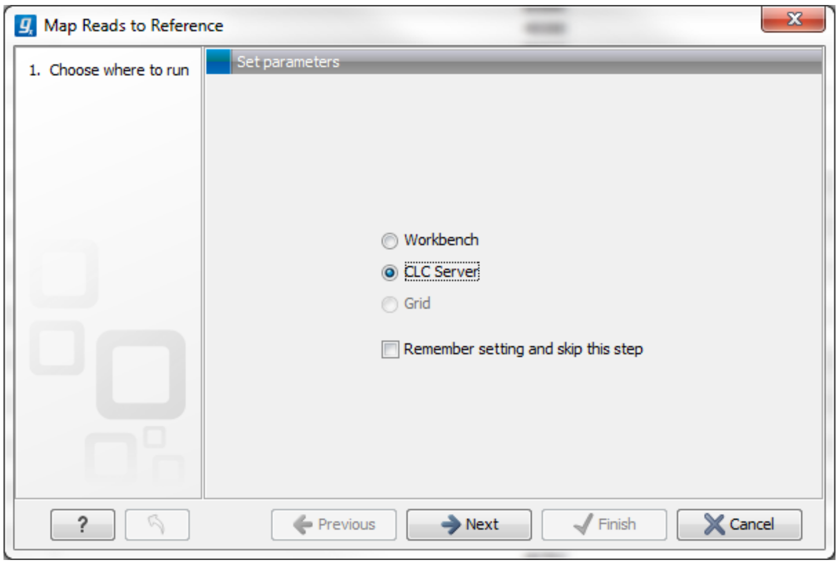


Figure 13.3: *A workflow is installed and ready to be used.*



Workbench [✓ CLC Server] Grid

☐ Remember setting and skip this step

Figure 13.4: *Selecting where to run the workflow.*

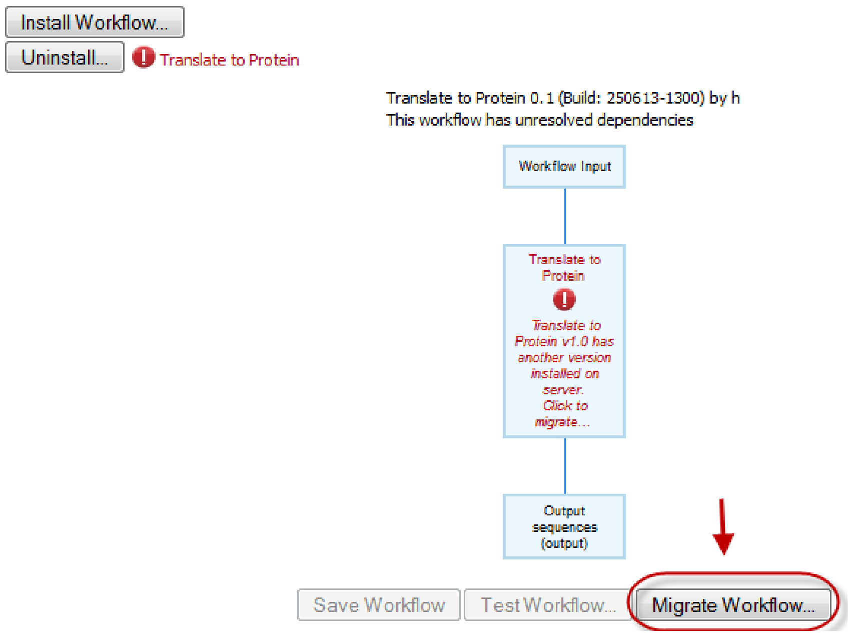
server as well as in the workbench. The only requirement is that both the tools that are part of the workflow and any reference data are available.

An important benefit of installing workflows on 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.

13.3 Automatic update of workflow elements

When new versions of the *CLC Server* are released, some of the tools that are part of a workflow may change. When this happens, the installed workflow may no longer be valid. If this is the case the workflow will be marked with an attention (❗) symbol.

When a workflow is opened in the administrative interface, a button labeled "Migrate Workflow" will appear whenever tools used in the workflow have been updated (see figure 13.5).



Translate to Protein 0.1 (Build: 250613-1300) by h
This workflow has unresolved dependencies

Translate to Protein v1.0 has another version installed on server. Click to migrate...

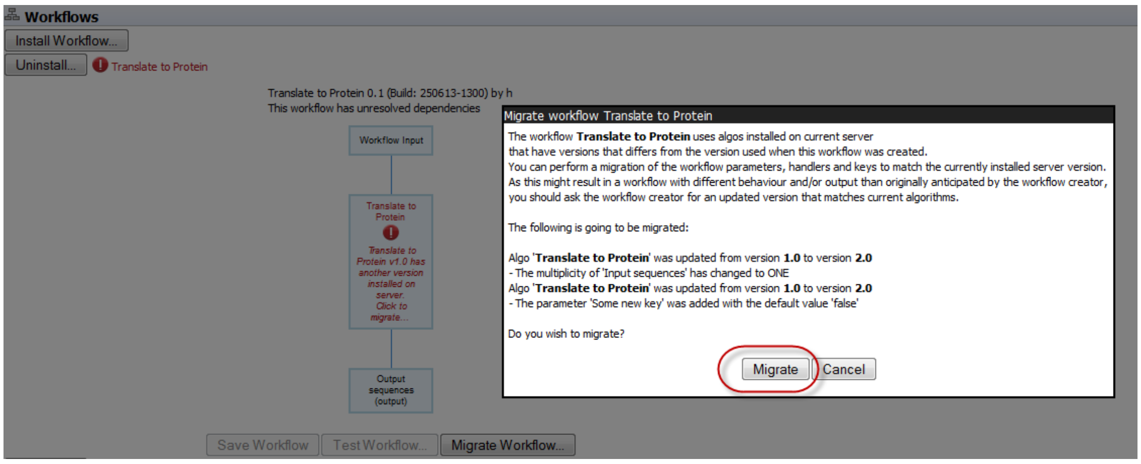
Figure 13.5: *When updates are available a button labeled "Migrate Workflow" appears with information about which tools should be updated. Press "Migrate Workflow" to update the workflow. The workflow must be updated to be able to run the workflow.*

Updating a workflow means that the tools in your workflow is updated with the most recent version of these particular tools. To update your workflow, press the **Migrate Workflow** button. This will bring up a pop-up dialog that contains information about the changes that you are about to accept. In case errors have occurred these will also be displayed here. The pop-up dialog allows to either accept the changes by pressing the "Migrate" button or cancel the update (figure 13.6).

Pressing "Migrate" will update the workflow, which then will be marked with a green check mark (✓). The updated workflow keeps the name of the original workflow. **Note!** In cases where new parameters have been added, these will be used with their default settings.

As there may be situations where it is important for you to keep the workflow in its original form, a copy is created of the original workflow with the original name extended with "-backup (disabled)". This is shown in figure 13.7.

When clicking on the copy of the original workflow, a button labeled "Re-enable Workflow" appears (figure 13.8). Pressing this button will re-enable the original workflow and uninstall the updated workflow.



Translate to Protein 0.1 (Build: 250613-1300) by h
This workflow has unresolved dependencies

Migrate workflow Translate to Protein

The workflow **Translate to Protein** uses algos installed on current server
that have versions that differs from the version used when this workflow was created.
You can perform a migration of the workflow parameters, handlers and keys to match the currently installed server version.
As this might result in a workflow with different behaviour and/or output than originally anticipated by the workflow creator,
you should ask the workflow creator for an updated version that matches current algorithms.

The following is going to be migrated:

Algo '**Translate to Protein**' was updated from version **1.0** to version **2.0**
- The multiplicity of 'Input sequences' has changed to ONE
Algo '**Translate to Protein**' was updated from version **1.0** to version **2.0**
- The parameter 'Some new key' was added with the default value 'false'

Do you wish to migrate?

Figure 13.6: *A pop-up box allows you to select whether you wish to update the workflow or not. Press "Migrate" to update the workflow.*

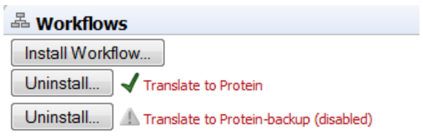
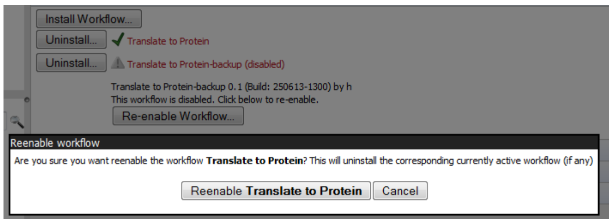


Figure 13.7: *In addition to the updated version of the workflow that now is marked with a green check mark, a copy is created of the original workflow with the original name extended with "-backup (disabled)".*



Translate to Protein-backup 0.1 (Build: 250613-1300) by h
This workflow is disabled. Click below to re-enable.

Reenable workflow

Are you sure you want reenable the workflow **Translate to Protein**? This will uninstall the corresponding currently active workflow (if any)

Figure 13.8: *After a workflow has been updated, it is possible to re-enable the original workflow.*

Chapter 14

Command line tools

CLC Server Command Line Tools is a command-line client for *CLC Server*. You can find a complete overview of usage for all commands in the manual found here (html version): http://resources.qiagenbioinformatics.com/manuals/clcservercommandlinetools/current/ or here (pdf version): http://resources.qiagenbioinformatics.com/manuals/clcservercomman current/User_Manual.pdf.

In the *CLC Server Command Line Tools* manual you can find information about how to download and install *CLC Server Command Line Tools*, the basic usage of the command line tools, and you can also find an example script that can be modified by the user.

Chapter 15

Appendix

15.1 Use of multi-core computers

Many tools in CLC Workbenches and Servers can make use of multi-core CPUs. This does not necessarily mean that all available CPU cores are used throughout the analysis. It means that these tools benefit from running on computers with multiple CPU cores.

Tools available differ between CLC Workbenches. In the table, the availability of these tools in different CLC Workbench Toolbox menus is indicated with an X.

Use of multi-core computers	Genomics	Drug Discovery	Biomedical Genomics
Add Conservation Scores			X
Add Information about Amino Acid Changes			X
Add Information from Variants			X
Add Exon Number			X
Add Flanking Sequence			X
QC for Read Mapping			X
Amino Acid Changes	X		
Annotate and Merge Counts	X		X
Annotate from Known Variants	X		
Annotate with Conservation Score	X		
Annotate with Exon Numbers	X		
Annotate with Flanking Sequences	X		
Basic Variant Detection	X		X
BLAST (will not scale well on many cores)	X		
Compare Sample Variant Tracks	X		
Copy Number Variant Detection			X
Create Alignment	X	X	X
Create Detailed Mapping Report	X		
Create Sequencing QC Report	X		
Create Statistics for Target Regions	X		
De Novo Assembly	X		
Dock Ligands		X	
Download Reference Genome Data	X		
Extract and Count	X		X
Filter against Control Reads	X		
Filter against Known Variants	X		
Filter Marginal Variant Calls	X		
Filter Reference Variants	X		
Fisher Exact Test	X		
Fixed Ploidy Variant Detection	X		X
GO Enrichment Analysis	X		
Identify Enriched Variants in Case vs Control Group			X
Identify Highly Mutated Gene Groups anc Pathways			X
Identify Variants with Effect on Splicing			X
Import Molecules from SMILES or 2D		X	
InDels and Structural Variants	X		X
K-mer Based Tree Construction	X		
Link Variants to 3D Protein Structure	X		X
Local Realignment	X		X
Low Frequency Variant Detection	X		X
Map Reads to Contigs	X		
Map Reads to Reference	X		X
Maximum Likelihood Phylogeny	X		
Merge Annotation Tracks	X		
Model Testing	X		
Predict Splice Site Effect	X		

Use of multi-core computers	Genomics	Drug Discovery	Biomedical Genomics
Probabilistic Variant Detection (legacy)	X		X
QC for Sequencing Reads			X
QC for Read Mapping			X
QC for Targeted Sequencing			X
Quality-based Variant Detection (legacy)	X		X
Remove False Positives			X
Remove Germline Variants			X
Remove Reference Variants			X
Remove Variants Found in Common dbSNP			X
Remove Variants Found in External Database			X
Remove Variants Found in HapMap			X
Remove Variants Found in 1000 Genomes Project			X
Remove Variants Inside Genome Regions			X
Remove Variants Not Found in External Database			X
Remove Variants Outside Genome Regions			X
Remove Variants Outside Targeted Regions			X
RNA-Seq Analysis	X		X
Screen Ligands		X	
Trim Sequences	X		X
Trio Analysis	X		X

15.2 Troubleshooting

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

15.2.1 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**.

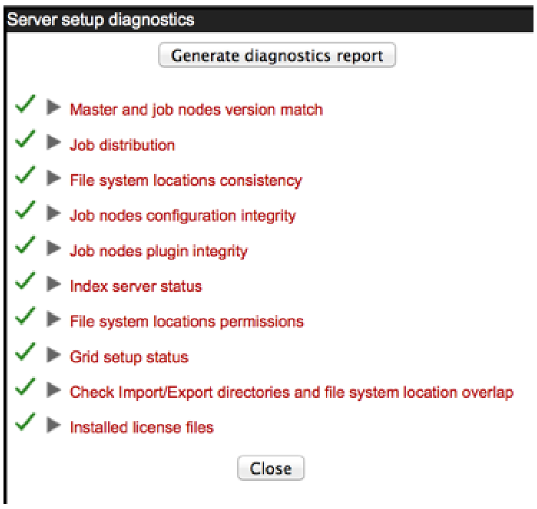
This will show a list of test that are performed on the system as shown in figure 15.1.

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.

15.2.2 Bug reporting

When contacting AdvancedGenomicsSupport@qiagen.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



Server setup diagnostics

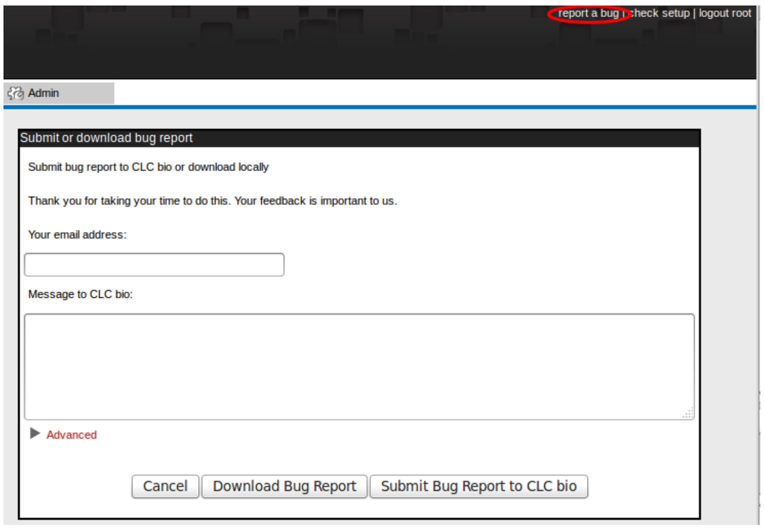
Generate diagnostics report

- ✓ Master and job nodes version match
- ✓ Job distribution
- ✓ File system locations consistency
- ✓ Job nodes configuration integrity
- ✓ Job nodes plugin integrity
- ✓ Index server status
- ✓ File system locations permissions
- ✓ Grid setup status
- ✓ Check Import/Export directories and file system location overlap
- ✓ Installed license files

Close

Figure 15.1: *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.*

You can see the bug report dialog in 15.2.



Submit or download bug report

Submit bug report to CLC bio or download locally

Thank you for taking your time to do this. Your feedback is important to us.

Your email address:

Message to CLC bio:

Advanced

Cancel | Download Bug Report | Submit Bug Report to CLC bio

Figure 15.2: *Submitting a bug report.*

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.

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 QIAGEN support. If the server has access to the internet, you can **Submit Bug Report**.

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 AdvancedGenomicsSupport@qiagen.com is treated confidentially.

No password information is included in the bug report.

15.3 Database configurations

For *CLC Server* solutions where the license includes the add-on *CLC Bioinformatics Database*, support for data management in an SQL-type database is available.

15.3.1 Getting and installing JDBC drivers

For MySQL or Oracle databases, the appropriate JDBC driver must be available for the application. If you do not already have the appropriate JDBC driver, it needs to be downloaded from the provider and then placed in the `userlib` directory in the installation area of the CLC software.

Details for the MySQL JDBC Driver

1. Go to the page `http://dev.mysql.com/downloads/connector/j/` to download the driver.
2. Please choose the option **Platform Independent** when selecting a platform.
3. After clicking on the button to Download, you can login if you already have an Oracle Web account, or you can just click on the link that says `No thanks, just start my download` further down the page.
4. Uncompress the downloaded file and move the driver file, which will have a name of this form: mysql-connector-java-X.X.XX-bin.jar, to the folder called `userlib`.

Details for the Oracle JDBC Driver

1. Go to the page `http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html`.
2. Select the version for your Oracle database version that will work with Java 1.7.

   For example, for 11g, the ojdbc6.jar includes classes for use with JDK 1.7.

   You will need an Oracle account to download the driver.
3. Move the driver jar file to the folder called `userlib`.

Completing the installation

After the JDBC driver is in the `userlib` folder, then:

- For a stand-alone Server instance, restart the Server software.

- For a CLC job node setup, the JDBC driver file must be placed in the `userlib` folder in the CLC software installation area on **the master node as well as each job node system**. The CLC software needs to be restarted after the driver is placed in this folder.
- If running a grid setup, the JDBC driver file is placed in the `userlib` folder in the CLC Server software installation area. After the driver file is in place, restart the Server software. This will deploy the changes to the grid workers.

15.3.2 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 an 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`

15.4 SSL and encryption

The *CLC Server* supports SSL communication between the Server and its clients (i.e. 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.

15.4.1 Enabling SSL on the server

A **server certificate** is required before SSL can be enabled on the *CLC 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 `pkcs12` 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 15.4.1).

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

Next, the `server.xml` file in the `conf` subdirectory of the *CLC 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 port number you have configured it to.

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.

15.4.2 Logging in using SSL from the Workbench

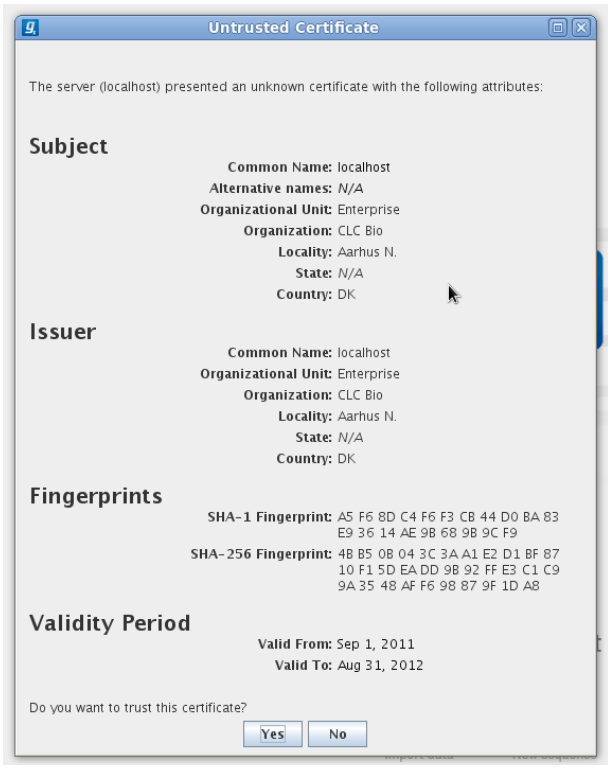
When the Workbench connects to the *CLC Server* it automatically detects if Secure Socket Layer (SSL) should be used on the port it is connecting to or not.

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 15.3.

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 15.4.

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



Untrusted Certificate

The server (localhost) presented an unknown 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

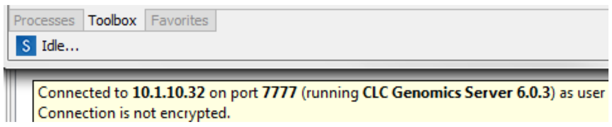
SHA-1 Fingerprint: A5 F6 8D C4 F6 F3 CB 44 D0 BA 83 E9 36 14 AE 9B 68 9B 9C F9
SHA-256 Fingerprint: 4B B5 0B 04 3C 3A A1 E2 D1 BF 87 10 F1 5D EA DD 9B 92 FF E3 C1 C9 9A 35 48 AF F6 98 87 9F 1D A8

Validity Period

Valid From: Sep 1, 2011
Valid To: Aug 31, 2012

Do you want to trust this certificate?

Figure 15.3: *A warning is shown when the certificate is not signed by a recognized CA.*



Connected to **10.1.10.32** on port **7777** (running **CLC Genomics Server 6.0.3**) as user
Connection is not encrypted.

Figure 15.4: *Showing details on the server connection by placing the mouse on the globe.*

15.4.3 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:

```
./clcserver -S localhost -U root -W default -P 8443
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 `clcsserversslstore` tool must be used:

```
./clcsserversslstore -S localhost -U root -W default -P 8443
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 `clcserver` program is allowed to connect to the server.

15.5 Non-exclusive Algorithms

Below is a list of algorithms which are non-exclusive, meaning that multiple of these algorithms can be run on one job or grid node.

Algorithms marked as *Streaming* are I/O intensive and two streaming algorithms will not be run at the same time. When running on grid, *Streaming* algorithms are treated as exclusive, meaning that they will never run in conjunction with other algorithms (or themselves). The two *Server types* are B = Biomedical Genomics Server Extension, and G = CLC Genomics Server.

Algorithm	Streaming	Server type
Add attB Sites		B, G
Add Conservation Scores	X	B
Add Exon Number	X	B
Add Flanking Sequence	X	B
Add Information about Amino Acid Changes	X	B
Add Information from Variant Databases	X	B
Amino Acid Changes	X	G
Annotate and Merge Counts		B, G
Annotate from Known Variants	X	G
Annotate with Conservation Score	X	G
Annotate with Exon Numbers	X	G
Annotate with Flanking Sequences	X	G
Annotate with Nearby Gene Information		B, G
Annotate with Overlap Information	X	G
Assemble Sequences		B, G
Assemble Sequences to Reference		B, G
BLAST at NCBI		B, G
ChIP-Seq Analysis		B, G
ChIP-Seq Analysis (legacy)		G
Compare Sample Variant Tracks	X	G
Convert DNA To RNA	X	G
Convert from Tracks	X	G

Algorithm	Streaming	Server type
Convert RNA to DNA	X	G
Convert to Tracks	X	G
Count-based statistical analysis		G
Coverage Analysis		G
Create Alignment		G
Create BLAST Database		G
Create Combined RNA-Seq Report		B, G
Create Detailed Mapping Report		G
Create Entry Clone (BP)		B, G
Create Expression Browser		B, G
Create Expression Clone (LR)		B, G
Create GC Content Graph Track		B, G
Create Histogram		B, G
Create Mapping Graph Tracks		G
Create New Genome Browser View		B
Create Statistics for Target Regions		G
Create Track List		G
Create Tree		G
Create Venn Diagram for RNA-Seq		B, G
Demultiplex Reads		B, G
Download 3D Protein Structure Database	X	B, G
Empirical Analysis of DGE		B, G
Extract and Count		B, G
Extract Annotations		G
Extract Consensus Sequence		G
Extract Reads Based on Overlap		B, G
Extract Sequences	X	B, G
Fasta High-Throughput Sequencing Import	X	G
Filter against Control Reads	X	G
Filter against Known Variants	X	G
Filter Annotations on Name	X	G
Filter Based on Overlap	X	G
Filter Marginal Variant Calls	X	G
Filter Reference Variants	X	G
Find Binding Sites and Create Fragments		B, G
Find Open Reading Frames		G
Fisher Exact Test	X	G
Gene Set Test		B, G
GO Enrichment Analysis		G
Identify Enriched Variants in Case vs Control Group	X	B
Identify Graph Threshold Areas		B, G
Identify Highly Mutated Gene Groups and Pathways		B
Identify Variants with Effect on Splicing		B
Illumina High-Throughput Sequencing Import	X	B, G
Import SAM/BAM Mapping Files	X	B, G
Import Tracks from File		B, G
InDels and Structural Variants		B, G

Algorithm	Streaming	Server type
Ion Torrent High-Throughput Sequencing Import	X	B, G
Link Variants to 3D Protein Structure	X	B, G
Merge Annotation Tracks	X	G
Merge Overlapping Pairs		B, G
Merge Read Mappings	X	B, G
Motif Search		G
Gaussian Statistical Analysis		G
PacBio High-Throughput Sequencing Import	X	B, G
Predict Splice Site Effect		G
Principal Component Analysis		B, G
QC for Read Mapping		B
QC for Target Sequencing		B
Remove False Positives	X	B
Remove Germline Variants	X	B
Remove Reference Variants	X	B
Reverse Complement Sequence		G
Reverse Sequence		G
Roche 454 High-Throughput Sequencing Import	X	B, G
Sanger High-Throughput Sequencing Import	X	B, G
Secondary Peak Calling		B, G
Select Genes by Name	X	B
Solid High-Throughput Sequencing Import	X	B, G
Translate to Protein		G
Trim Sequences		B, G
TRIO analysis		B, G
Whole Genome Coverage Analysis		B

15.6 DRMAA libraries

Distributed Resource Management Application API (DRMAA) libraries are provided by third parties. Please refer to the distributions for instructions for compilation and installation. CLC bio cannot troubleshoot nor provide support for issues with DRMAA libraries themselves. Please refer to the DRMAA providers if issues associated with building and installation occur. Information in this section of the manual is provided as a courtesy, but should not be considered a replacement for reading the documentation that accompanies the DRMAA distribution itself.

15.6.1 DRMAA for LSF

The source code for this library can be downloaded from `https://github.com/PlatformLSF/lsf-drmaa`. Please refer to the documentation that comes with the distribution for full instructions. Of particular note are the configure parameters "–with-lsf-inc" and "–with-lsf-lib" parameters, used to specify the path to LSF header files and libraries respectively.

15.6.2 DRMAA for PBS Pro

Source code for this library can be downloaded from `http://sourceforge.net/projects/pbspro-drmaa/`.

Please refer to the documentation that comes with the distribution for full instructions. Of particular note is the inclusion of the "–with-pbs=" parameter, used to specify the path to 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 area, along with other files.

Please note that 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 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.

15.6.3 DRMAA for OGE or SGE

OGE/SGE comes with a DRMAA library.

15.7 Consumable Resources

Setting up Consumable Resources with LSF

The following information was provided by IBM.

If you have questions or issues with setting up a consumable resource for LSF, please refer to your LSF documentation. For questions not covered there, please contact ruzhuchen@us.ibm.com and achristi@ca.ibm.com.

LSF has the ability to do "license scheduling" and ensure that CLC Server jobs running under LSF are only dispatched when there are available CLC Grid Worker licenses. When such scheduling is configured, CLC jobs for which no free licenses are available would stay in "pend" status, waiting for a CLC Grid Worker license to become available.

There are two parts to making use of this type of scheduling:

1. Configure the consumable resource in LSF.
2. Specify a clcbio license reservation when jobs are submitted to LSF.

Configuring the consumable resource in LSF

Add a consumable resource called clcbio in $LSF_ENVDIR/lsf.shared:

```
Begin Resource
RESOURCENAME  TYPE  INTERVAL INCREASING DESCRIPTION # Keywords
    mips      Boolean ()       ()       (MIPS architecture)
...
...
    clcbio    Numeric ()        N       (clcbio license)
 End Resource
```

Add the number of clcbio licenses in $LSF_ENVDIR/lsf.cluster.<clustername>:

```
Begin ResourceMap
 RESOURCENAME  LOCATION
 # CLCBIO license resource
 clcbio        (14@[all])  # 14 clcbio licenses can be used
 #....
End ResourceMap
```

This example shows a configuration for 14 CLC Grid Worker licenses, which means that up to 14 CLC jobs can be running on the LSF cluster at the same time. This integer needs to be changed to the number of licenses you own.

The configuration shown here assumes the CLC Grid Worker licenses can only be use in the LSF cluster as LSF will manage the free token count from the scheduling side.

In this context, LSF does not replace or directly talk to the LMX license server for CLC licenses. Rather, LSF manages the CLC Grid Worker license reservations internally.

Specify a clcbio license reservation when jobs are submitted to LSF

CLC jobs submitted to LSF need to have a clcbio license reservation specified.

This can be done in several different ways:

- via the CLC Grid Preset "Native Specification" field. (This is the most convenient method.) Simply add:

  ```
  -R "rusage[clcbio=1]"
  ```

  to this field.
- via the batch job submission command line
- using the RES_REQ line inside the lsb.queues file
- via an application profile (lsb.applications)

Important: After any LSF configuration file changes, one needs to reconfigure LSF for the changes to take effect. That is, run:

```
lsadmin reconfig
badmin reconfig
```

These are "safe" commands to run. That is, pending LSF jobs will continue to "pend" in status and running LSF jobs will continue to run.

15.8 Third party libraries

The *CLC Server* includes a number of third party libraries.

Please consult the files named NOTICE and LICENSE in the server installation directory for the legal notices and acknowledgements of use.

For the code found in this product that is subject to the Lesser General Public License (LGPL) you can receive a copy of the corresponding source code by sending a request to our support team at AdvancedGenomicsSupport@qiagen.com.

15.9 External network connections

Some functionality available in CLC software requires access to specific addresses on the Internet. A list of the internet sites accessed and a listing of the tools involved can be found at `https://secure.clcbio.com/helpspot/index.php?pg=kb.page&id=242`. The list of sites there can be referred to when configuring firewall settings for networks that utilize a whitelist approach. For CLC Servers with nodes, any nodes that need to execute functionality listed on that webpage will need access to the relevant sites.

15.9.1 Proxy settings

To add proxy settings to the CLC server, you will need to add lines to the server `.vmoptions` file. This file can be found in the installation area of the CLC server software. The name will reflect the specific product. For example, for the CLC Genomics Server, it would be called `CLCGenomicsServer.vmoptions`.

The following lines must be added to that file:

```
-Dhttp.proxyHost=
-Dhttp.proxyPort=
-Dhttp.nonProxyHosts="localhost|127.0.0.1|11...|.foo.com|etc"
```

where the `proxyHost` IP address and the `proxyPort` port number need to be added to complete the top two lines.

For job nodes, the settings described here must be applied to each node that will need to submit jobs that need access to the internet. For the `nonProxyHosts` values in this case, specify the names or IP addresses/subnets for *all* job nodes (if applicable with your server configuration) that should not be using the proxy service, if relevant.

For grid node setups, a `clcgridworker.vmoptions` file must be created in each deployed gridworker area, if such a file does not already exist and the settings described above added. See also section 6.3.10.

15.10 Monitoring

We recommend that you monitor the health and performance of the servers that the *CLC Server* software is running on and also monitor some key metrics of the *CLC Server* itself. This will enable you to react quickly to any problems that occur and can aid in optimization of server performance.

With regards to the servers' physical resources, monitoring the amount of free memory and disk space is recommended, as consumption of these can be substantial depending on types and numbers of analyses being run.

Monitoring metrics of the *CLC Server* itself enables you to keep tabs on how well the jobs processing is going. The metrics the *CLC Server* provides are available as JMX[1] attributes. Software from a third party will be necessary to set up the monitoring of these attributes. Numerous software products for this are available and most support JMX. If your monitoring software supports the raising of alarms, you can set up triggers based on these metrics to receive alerts when a situation arises that needs attention.

15.10.1 Setting up JMX monitoring

No special configuration is necessary if monitoring will take place locally. However, JMX must be enabled for remote monitoring. This is done by adding a few settings to the server vmoptions file. The relevant `.vmoptions` file is located in the root of the server installation folder of a single server, or the root of the installation folder of the master server in the case of a node setup (Hereafter this folder will be referred to as `CLC_SERVER_BASE`).

Enable JMX with no security Add the following to the `.vmoptions` file:

```
-Dcom.sun.management.jmxremote.port=9999
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.authenticate=false
```

Enable JMX with authentication

1. Add the following to the .vmoptions file, instead of the lines provided in the section above.

   ```
   -Dcom.sun.management.jmxremote.port=9999
   -Dcom.sun.management.jmxremote.ssl=true
   -Dcom.sun.management.jmxremote.authenticate=true
   -Dcom.sun.management.jmxremote.password.file=../conf/jmxremote.password
   -Dcom.sun.management.jmxremote.access.file=../conf/jmxremote.access
   ```

2. Create the access authorization file `CLC_SERVER_BASE/conf/jmxremote.access` and write the following in it:

   ```
   monitorRole readonly
   controlRole readwrite
   ```

3. Create the password file `CLC_SERVER_BASE/conf/jmxremote.password` and write the following in it:

[1] https://en.wikipedia.org/wiki/Java_Management_Extensions

```
monitorRole clcserver
controlRole clcserver
```

Further information about setting up JMX monitoring can be found in the Oracle guide on Monitoring and Management Using JMX Technology `http://docs.oracle.com/javase/8/docs/technotes/guides/management/agent.html`.

15.10.2 Completed process metrics

Object name: `com.clcbio.server:type=CompletedProcesses`

The "completed process" metrics can be used to evaluate the processes that are complete either because they were successful or because they failed. Canceled processes are ignored. The *CLC server* provides a temporal view of the latest completed processes, with two different temporal views available: time frame view or a history view that includes a fixed number of the most recently completed processes.

The size of the time frame to view and the number of entries for the history view can be configured directly through JMX or by editing the `Monitoring.properties` file. This properties file is located in this folder: `CLC_SERVER_BASE/settings/`. Please change the default settings to values that fit your specific setup. Lower values will generally result in a more reactive monitoring solution, but values that are too low may result lead to false alarms.

The following is a list of all the process related metrics that are available:

Number of processes in history

Attribute name: `NumberOfProcessesInHistory`

The number of processes currently in the history. Successful and failed processes are included. Canceled processes are not included. The maximum size of the history can be set using the SizeOfHistory attribute, available through JMX and the configuration file.

Number of failed processes in history

Attribute name: `NumberOfFailedProcessesInHistory`

The number of failed processes currently in the history.

Fraction of failed processes in history

Attribute name: `FractionOfFailedProcessesInHistory`

The fraction of the processes in the history that have failed. This fraction is not of much value if the number of processes in the history is very low.

Number of processes within time frame

Attribute name: `NumberOfProcessesWithinTimeFrame`

The number of processes that have been completed between now and a variable number of milliseconds earlier. Both successful and failed processes are included. Canceled processes are not included. The time frame can be set using the TimeFrameInMilliseconds attribute, that is available through JMX and the configuration file.

Number of failed processes within time frame

Attribute name: `NumberOfFailedProcessesWithinTimeFrame`

The number of failed processes that have been completed between now and a variable number of milliseconds earlier.

Fraction of failed processes within time frame

Attribute name: `FractionOfFailedProcessesWithinTimeFrame`

The fraction of failed processes compared to the total number of processes that have been completed between now and a variable number of milliseconds from now. This fraction is not of much value if the number of processes in the time frame is very low.

15.10.3 Process execution metrics

Object name: `com.clcbio.server:type=ProcessExecution`

Process execution metrics allow measurement of how many jobs are being processed and how many are in a queue because they are waiting for available processing resources.

On job node setups, the object names are available on all nodes, but it only makes sence to monitor them on the master node since this is where the jobs are managed. If a grid is used to process the jobs, the actual queue will be a part of the grid system, which results in the processes being moved almost instantly to the "currently processing" state and the *CLC Server* queue itself is then empty.

Currently processing

Attribute name: `CurrentlyProcessing`

Number of processes being executed at the moment.

Waiting for resources

Attribute name: `WaitingForResources`

On a job node setup, the umber of processes that are queued and are waiting for a node to be available for processing. This does not include processes that are waiting for output from another process. It includes only processes that have the input they need and are ready to be processed, but where no resources are available at that moment.

15.10.4 Job node metrics

Object name:
```
com.clcbio.server:type=JobNodes
com.clcbio.server:type=JobNodes,name=<job node name>
```

With the job node metrics you can monitor a master node's connection with its job nodes. The object name is available on all nodes, but it only makes sence to monitor this on the master node. There are two sets of attributes to monitor. One set provides an aggregated view of all the job nodes while the other provides individual attributes for each job node.

Communication errors are only reported if the server uses the General queue. If the Hight throughput queue is used, the master node never contacts the job nodes on its own initiative and therefore there is no support for monitoring the job nodes through JMX in the current version of the server.

Apart from communication error attributes, the set of individual attributes also includes information about the host and port of a given job node. This information is not meant for monitoring as such, but is included for convenience, as when an error does arise identification of the job node involved is usually necessary.

Communication failed

Attribute name: `CommunicationFailed`

This attribute is set to true if the master node is currently having problems communicating with this particular job node.

Seconds since communication failed

Attribute name: `SecondsSinceCommunicationFailed`

The number of seconds since the communication with the job node first failed. As soon as the master node succeeds in connecting with the job node again, this value returns to zero. This attribute can be useful if you want to avoid reacting to very short fallouts in communication.

Max seconds since communication failed

Attribute name: `MaxSecondsSinceCommunicationFailed`

The maximum number of seconds since communication with any of the job nodes first failed. The advantage of this metric is that monitoring of it can be set up once and does not need to be changed if a job node is attached or detached.

Number of job nodes

Attribute name: `NumberOfJobNodes`

The number of job nodes currently attached to the master server.

Number of failed job nodes

Attribute name: `NumberOfFailedJobNodes`

The number of job nodes the master server currently has problems communicating with.

Bibliography

[Langmead et al., 2009] Langmead, B., Trapnell, C., Pop, M., and Salzberg, S. L. (2009). Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome Biol*, 10(3):R25.

[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