CLC Server Command Line Tools

User manual

Manual for

CLC Server Command Line Tools 1.7

Windows, Mac OS X and Linux

April 16, 2013

This software is for research purposes only.

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

Contents

Chapter 1

Introduction

Welcome to the user manual of *CLC Server Command Line Tools*.

The *CLC Server Command Line Tools* is a command-line client for the CLC Genomics Server[1]. The *CLC Server Command Line Tools*, and in particular provide the tools to start analyses and other tasks on CLC Servers, including data import and export, and utility data operations such as moving, renaming and deleting data on the server.

A typical work flow using the *CLC Server Command Line Tools* might be:

1. Import your sequence data
2. Run analyses such as read mapping, SNP detection or RNA-Seq
3. (Optionally) export the results to your local disk

Another client available to run tasks on the CLC Genomics Server is the graphical *CLC Genomics Workbench*. Below are recommendations for choosing which of these two clients, the graphical or the command line, to use for your work:

- For *visualization and interpretation of data* we recommend the *CLC Genomics Workbench*. The only way to visualize and interpret data when you have worked using the *CLC Server Command Line Tools* is to export the results into file formats that can be imported into visualization tools.
- For *explorative work* we recommend using the *CLC Genomics Workbench*. The numerous parameters are easier to interpret using the graphical interface, and selection and management of data is more intuitive through this interface for most users. In addition, the graphical user interface has more constraints to help guide reasonable choices of parameters and combination of parameters; these constraints are not all present in the *CLC Server Command Line Tools*.
- For automation and consistency, of particular utility in production environments, the *CLC Server Command Line Tools* client is recommended. In particular, you can *script pipelines*

[1] Like any other client software, the *CLC Server Command Line Tools* would most commonly be installed and used on systems other than the one that the CLC Server software is installed on, although there is no restriction meaning that this must be the case.

of analyses on the CLC Genomics Server, and then use these scripts for processing many data sets in a consistent manner. For initial pilot runs, it is often helpful to use the exploratory features of the *CLC Genomics Workbench* to determine quality control and parameter settings, and then incorporate these settings into a script using the *CLC Server Command Line Tools*.

This user manual begins with installation instructions followed by an explanation of the basics of operating the *CLC Server Command Line Tools*. Then, we provide an example script, which illustrates various aspects of how to use the analysis tools available on the server.

1.1 Installation

The *CLC Server Command Line Tools* can be downloaded from http://www.clcbio.com/products/clc-server-command-line-tools-direct-download and is available for Windows, Mac and Linux. You can install the tools on any computer that can connect to your CLC Genomics Server, but it makes sense to install them onto the computer that will be used to run the scripts, or onto the server computer itself.

1.2 System requirements

The system requirements of *CLC Server Command Line Tools* are these:

- Windows XP, Windows Vista, Windows 7, Windows 8, Windows Server 2003 or Windows Server 2008
- Mac OS X 10.6 or later. However, Mac OS X 10.5.8 is supported on 64-bit Intel systems.
- Linux: Red Hat 5 or later. SUSE 10 or later.
- 32 or 64 bit
- 256 MB RAM required
- 512 MB RAM recommended
- 1024 x 768 display recommended

You will also need a running version of *CLC Genomics Server*. No additional license is required for running the *CLC Server Command Line Tools*.

Chapter 2

Basic usage

Once installed, there will be three programs present in the installation folder:

- `clcserver` - the key program. It is used to run all the commands that communicate with the server.
- `clcresultparser` - used to parse data locations from particular text files generated during clcserver runs. This command is most useful when connecting analyses in a scripting pipeline (see section 2.3).
- `clcserverkeystore` - a helper tool for enabling passwords to be handled securely (see section 2.1).

The `clcserver` program requires the following four flags, which provide information about the connection to the server:

–S <hostname or IP address of the server>

–P <port the server runs on> When omitted, port 7777 is used, which is the default for server installations.

–U <user name> The username used to log into the server.

–W <password or token> See section 2.1 for how to avoid entering passwords in clear text.

If you run the clcserver command with the above parameters, and nothing else, then a list of all commands that can be run on the server will be returned. For example:

```
clcserver –S server.com –U bob –W secret
```

The commands to be run on the server are supplied with the flag:

–A <command to be executed on server>

If you supply the -A flag with a program name, but do not provide the required flags for that program, then a listing of the flags for that program will be returned. For example, a command of a form like:

```
clcserver -S server.com -U bob -W secret -A read_mapping
```

would return the full list of parameters for the `read_mapping` function, including the possible values, and descriptions. This information, for each command, is also available in the online manual at http://www.clcsupport.com/clcservercommandlinetools/current/index.php?manual=Usage_all_commands.html).

An optional flag when working on the command line, but important when working with scripts, is:

–O <filename> The name of a file to be created to hold a summary of steps carried out on the server and data locations of the results generated. The data locations are of a form that can be used by downstream CLC commands. See section 2.3 for information about parsing this file. By default, this file is placed in your working directory. If you do not provide this flag, this data will be written to a file called `results.txt`

For those working with the *CLC Grid Integration Tool*, you can run import and algorithm commands through your grid nodes by adding the following flag to your clcserver command:

–G <grid preset name>

Other optional flags available for the clcserver command are:

–C <integer> Specify the **column width** of the help output.

–D <boolean> Enables **debug** mode when set to true, providing more elaborate output and error messages.

–H Display general **help** instructions.

–V Display the **version number** of *CLC Server Command Line Tools*.

2.1 Handling passwords

To help you avoid sending your server login password in clear text across the network, we provide the `clcserverkeystore` tool. This enables you to convert your password to a token, which is stored and can be interpreted by the *CLC Server Command Line Tools* when logging onto the server. The token is encrypted and saved with the user profile on the computer running the *CLC Server Command Line Tools*.

You can generate a password token using the following command:

```
clcserverkeystore --generate
```

You will be prompted for the password. After you have typed the password, press the **Enter** key. The password token is then returned on screen. It will be a long string of text that you should save somewhere to refer to for future use.

So, if we say that user `bob` has password `secret`, and has generated a password token `CAIHMAAAAAAAAAPcb769377f4`, then he could enter either of the following two commands to connect to his server. The first passes the password in plain text. The second, passes it as an encrypted token.

```
clcserver -S server.com -U bob -W secret
```

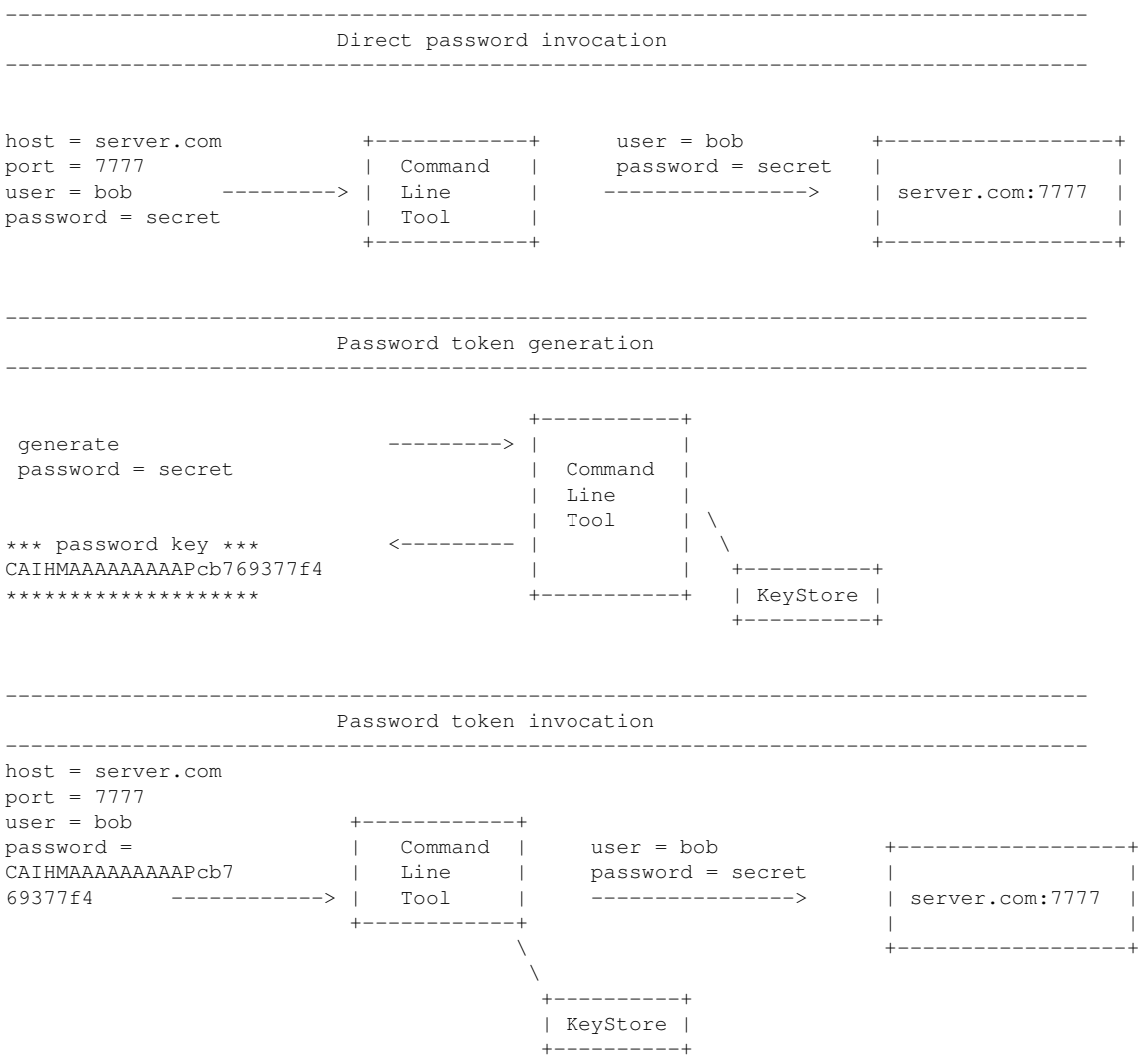
```
clcserver -S server.com -U bob -W CAIHMAAAAAAAAAPcb769377f4
```

If the token needs to be deleted, the `clcserverkeystore` program has two other parameters that can be used:

-d <token> This will delete the individual token provided as a parameter.

deleteAll This will delete all the tokens in the user profile.

The first section of the diagram below illustrates the process of logging into the server using a clear text password. The second section illustrates the process of generating a password token and storing it in the keystore, followed by a section showing how the token is substituted by the *CLC Server Command Line Tools* with the real password when initiating the connection to the server.



2.2 Data objects, data files and the CLC URL

In this section, we refer to data already in a CLC Server data area as being in a `persistence model`. This technical term allows us to refer to any area that the CLC Servers or Workbenches recognize as CLC data areas. Most relevant to this document are CLC Server File Locations and Database Locations. Each Server File Location, for example, is a single persistence model.

When running the `clcserver` command, one needs to be able to specify resources inside of or outside of a CLC Server persistence model, as well as files and folders on the local machine.

Files residing on the local system are indicated by using the full or relative path to the file.

Data objects held in a CLC Server persistence model and files stored in areas configured as Import/Export locations for the CLC Server are indicated using a CLC URL. CLC data objects within a persistence model can be identified by using two different URL forms, one based on the object's **name** or using its **object ID**. For files in an Import/Export area, only the name-based URL form can be used.

2.2.1 The CLC URL - the ID form

Data resources within persistence models can be referred to using the object-ID form of CLC URLs. These look soomething like the following:

clc://node04:7777/3123-2131uafda-sads/213-sddsa123-5232

Getting the object ID form of a CLC URL There are several ways this can be done:

1. Via the Workbench.

   Copy the CLC URL by highlighting the data object by clicking on the object in the Workbench Navigation area to select it, and then using the keyboard short cut Ctrl-C. Then use Ctrl-V to paste the URL into a shell window, text editor or similar. See figure 2.1.

2. Using the CLC Server web administrative interface.

   Select a data object from the tree browser on the left hand side of the browser window, and then select the "Element info" tab in the main area of the browser window. Click on the link to CLC-URL. This shows two versions of the CLC URLÆ one using the name and one using the object ID.

3. Take the object ID from within the text ouptut file generated using the -O flag of the `clcserver` command.

   This would be the common route when running a series of commands via a script.

Benefits: The ID form of a CLC URL is impervious to changes to the name of a data object or the folders the data resides in. That is, such changes do not affect a data object's ID.

Drawbacks: They are unreadable by humans.

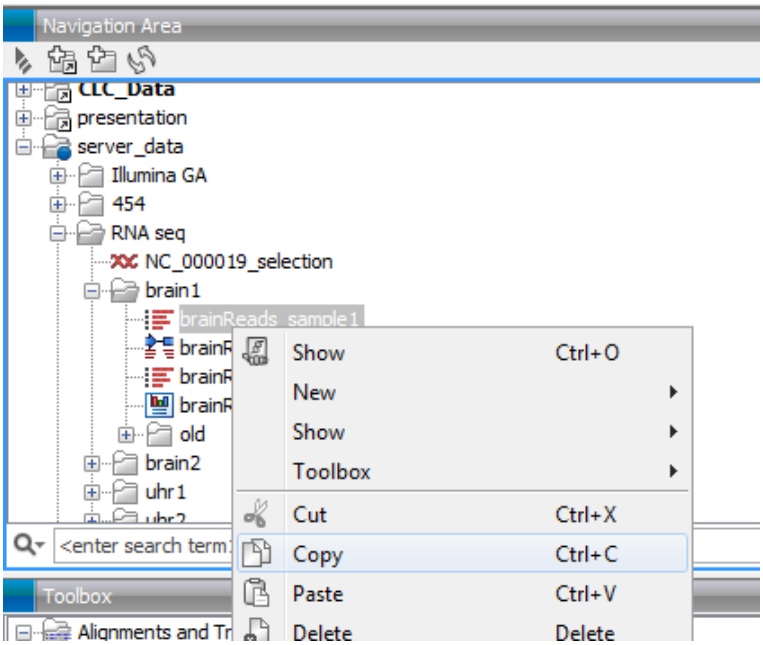


Figure 2.1: *Copying a data object in the workbench will put the CLC URL on the clipboard. You can then paste the URL into your command in the terminal.*

2.2.2 The CLC URL - name form

The object-name form of CLC URLs can be used to refer to data resources within persistence models or to refer to files located on the machine the CLC Server software is running on.

The first section of a object-name form of a CLC URL indicates whether it is referring to a **data object** in a CLC Server persistence model or to a **file** stored in an area configured as an Import/Export location for the CLC Server. These forms are:

clc://server - refers to a data object present in a persistence model. This part of the URL is then followed by the name of the persistence model the data is located in. For example, the name of particular CLC Server File Location. An example of this form is: `clc://server/CLC_Server_Project/alignments/myAlignment`

Note that for CLC data in a persistence model, you need the **name of the data object** (as seen via the Navigation area of the Workbench) **not** the name of the file holding the data object (as seen when listing files using system tools like the commands ls or dir).

clc://severfile - refers to a file in an area configured as an Import/Export area for the CLC Server. This form would commonly be used to point at files containing data that is about to imported into the CLC Server, or to indicate a location to export data to. An example of this form is:

`clc://serverfile/mnt/data/project1/s_1_1.sequence.txt`

Benefits: Human readable and easier for many people when first starting out working with the Command Line Tools or when just running a few commands directly (as opposed to via a script).

Drawbacks: Any changes to the names of data objects or folders in the persistence model will break the URL.

2.2.3 Indicating local system files or folders

When importing from a file or files on the local system (i.e. the machinethe `clcserver` command is being run on), or exporting to the local system, then just the *relative or full path* to the relevant file or directory can be provided. An example of such a path would be:

```
/home/username/somefolder/datafile.gb
```

In the case where the `clcserver` command is on the same machine that the CLC Server software is installed on, one could choose whether to consider a file as "local" or as a file that the CLC Server software has access to via an Import/Export location. In other words, one would have the choice whether to provide just the path, or the clc://serverfile URL form form to identify files on the server machine. A key difference here would be that the first would allow for any file on the system (for which one has appropriate privileges) to be indicated. The second allows only for access to files in areas configured as Import/Export data locations for the CLC Server.

2.3 Result files and connecting analyses in pipelines

For each run of `clcserver`, text information is returned providing a summary of the steps taken, and the locations, in ID form, of any files generated. The file containing this information will, by default, be created in the *current directory* and will be called `result.txt`. You can use the `-O` option for the `clcserver` command if you wish to specify an alternative file to be written to.

An example of contents in a typical result file is shown below. In this case the file that was generated after running the the trim algorithm using a sequence list called `reads` as input. The result file lists the three files that were produced.

```
//
Name: reads trimmed
ClcUrl: clc://127.0.0.1:7777/-268177574-YCAAAAAAAAAAAAPc673b0db8c5e724f--5d66a991-12d75090d93--7fff
//
//
Name: reads report
ClcUrl: clc://127.0.0.1:7777/-268177574-ADAAAAAAAAAAAAPc673b0db8c5e724f--5d66a991-12d75090d93--7fff
//
//
Name: Trim Sequences log
ClcUrl: clc://127.0.0.1:7777/-268177574-CAAAAAAAAAAAAAPc673b0db8c5e724f--5d66a991-12d75090d93--7fff
//
```

When creating pipelines stitching together several analyses, you parse the result file to get the location of the data produced, which is needed as input for the next algorithm.

The result file is just a text file, but it can still be a challenge to parse it to get the necessary CLC URLs. Thus, we provide a tool called `clc_result_parser` to help with this. It searches the result file for a text expression you provide, and returns the CLC URL for files where a match to that text has been found in the `Name:` field. The `Name` field will contain the name of the input data along with a description of the type of data held in that file location.

In the case above, you would probably search for the trimmed reads to use for further analysis, which could be done with a command like this:

```
clcresultparser -f result.txt -c trimmed
```

Here, the following text would be returned:

```
clc://127.0.0.1:7777/-268177574-YCAAAAAAAAAAAAPc673b0db8c5e724f--5d66a991-12d75090d93--7fff
```

The options for the `clcresultparser` program are:

–f <name of result file to parse> This option is required.

–c <text to search for> Text to search for in the `Name` field of the result file. If nothing is found, the exit code is 1.

–n <text that should not match> Text that should **not be contained** in the `Name` field of the result file.

–r <regexp> A **Java regular expression** used for matching the name of the output (see http://java.sun.com/docs/books/tutorial/essential/regex/index.html).

– –ignorelogs <boolean> By default, all analyses produce log files. You can provide `false` as the argument to this option to stop log files from being returned. This is equivalent to excluding all names ending with `log`, or `log` with a number suffix. The latter are generated when there is more than one log file in the same folder.

–p <prefix text> When more than one match is found, the data locations for all matches will be output as a space-separated list. By supplying a **prefix** string, you can stipulate what character(s) to separate the list using. E.g. If you need to send several files output by the `clcresultparser` command as arguments to `-i` options for the next analysis, simply provide `"-i"`as as the argument for the `-p` flag.

–e <integer> The number of CLC URLs that are **expected** to be returned. If this is not the number of results files that match the search string. the command will return with exit code 10. This option is designed for use in scripts where you will wish to carry out validation steps are you proceed through the pipeline. (On the command line, you check the error code returned by the previous command by typing `echo $?`.

–C <integer> Specifies the **column width** of the help output.

2.4 Executing workflows

It is possible to execute workflows installed on the server. Workflows are described in detail in the user manual of of *CLC Genomics Workbench* and *CLC Genomics Server* at http://www.clcbio.com/usermanuals.

Executing workflows is similar to executing algorithms, and the installed workflows will be listed when the `-A` is omitted. Parameters that are open for change on execution are displayed when the workflow is specified for the `-A` option. Please note that the parameter names have name of the workflow element pre-pended to make sure they are always unique.

2.5 Emptying the recycling bin for a CLC Server File Location

Each CLC Server File Location has a recycling bin, where files that users delete are put. Only members of the administrator group, as defined on the Genomics Server, can empty the recycle bin associated with Genomics Server file locations. This is because the recycle bin is a shared location for any given Genomics Server file location and many sites do not want all users to be able to access it directly, that is to be able to view things or delete other people's data.

One can avoid the need to periodically go in and manually empty recycling bins by setting up a script that is run as a cronjob, which includes a command of the following form:

```
clcserver -S <serverinfo> -P <portnumber> -U <adminusername> -W <password or tok
```

Above, `$LOCATIONNAME` would be replaced byt the name of the CLC Server File Location you wish to empty the recycling bin of.

Chapter 3

Example script

In this section, we present an example script showing a typical work flow consisting of the following steps:

- Import of NGS data
- Read mapping
- Variant detection

The result of the variant detection is then exported to the local file system in Excel format.

The script is intended only as an example. Hopefully it will be possible for you to modify it to fit your purposes.

You can download the script including data from
http://www.clcbio.com/files/CLCServerCommandLineTools/1.7/example-workflow.zip.

This is a shell script that will run on Linux and Mac OS.

```
#!/bin/bash
##############################################################################################
## Example workflow script for CLC Server Command Line Tools 1.6
## CLC bio, June 2012
##
## For full documentation please visit: http://clcbio.com/usermanuals
##############################################################################################


### SETTINGS (Edit before use) ###############################################################

# 1. Configure your server connection parameters

SERVER="localhost"
PORT="7777"
USER="root"
PASSWORD="default"


# 2. Configure the path to the CLC Server Command Line tools

SERVERCMDPATH="/home/user/clcservercmdline/clcserver"
PARSECMDPATH="/home/user/clcservercmdline/clcresultparser"
```

```
# 3. Configure High-throughput sequencing data import location for import data
#  - Use the server web-interface to setup a data import location
#  - Copy the example files from the data directory to this location
#  - Edit the IMPORTPATH variable below to a CLC URL point to this location
#
# For more info please visit:
# http://www.clcsupport.com/clcgenomicsserver/current/index.php?manual=Accessing_files_on_writing_to_
# areas_server_filesystem.html
#
# For information about CLC URLs please visit:
# http://www.clcsupport.com/clcservercommandlinetools/current/index.php?manual=Referring_files_CLC_URL.html

IMPORTPATH="clc://serverfile/tmp/cmdline"


# 4. Configure data paths for saving results
#  - Use the server web-interface to configure a file system location for data storage
#  - Edit the DATAPATH variable below to a CLC URL pointing to this location
#
# For more info please visit:
# http://www.clcsupport.com/clcgenomicsserver/current/index.php?manual=Adding_file_system_location.html
#
# For information about CLC URLs please visit:
# http://www.clcsupport.com/clcservercommandlinetools/current/index.php?manual=Referring_files_CLC_URL.html

DATAPATH="clc://server/test"
DIR="clc-example"
SUBDIR="workflow-example"



### FUNCTIONS ###################################################################################

function check_return_code {
return_code=$?
cmdname=$1
#echo Return code: $return_code
if [ $return_code -ne 0 ]
then
echo ""
echo "### Error during: $cmdname ###"
echo "Terminating script"
exit 1
fi
}

### COMMANDS ####################################################################################
SERVERCMD="$SERVERCMDPATH -S $SERVER -P $PORT -U $USER -W $PASSWORD"
PARSECMD=$PARSECMDPATH


### WORKFLOW SCRIPT #############################################################################

# Make a directory for DIR
echo ""
echo "Make a directory: $DIR"
$SERVERCMD -A mkdir -t ${DATAPATH} -n ${DIR} -O tmpdir_result.txt
check_return_code "make dir"
tmpdir=`$PARSECMD -f "tmpdir_result.txt" -c "clc-example"`
check_return_code "make dir result parsing"
rm tmpdir_result.txt


# Make subdirectory in DIR folder for result and data
echo ""
echo "Make subdirectory: $SUBDIR"
$SERVERCMD -A mkdir -t ${tmpdir} -n ${SUBDIR} -O mkdir_result.txt
check_return_code "Make sub dir"
destdir=`$PARSECMD -f mkdir_result.txt -p "-d" -c $SUBDIR`
```

```
check_return_code "Make sub dir result parsing"
rm mkdir_result.txt


# Import solid data
echo ""
echo "Import solid data"
$SERVERCMD -A ngs_import_solid -f "$IMPORTPATH/solid_matepair_F3.csfasta" -f "$IMPORTPATH/solid_matepair_F3._Q
check_return_code "Import solid data"
soliddata=`$PARSECMD -f ngs_import_solid_result.txt -p "-i" -c "solid" --ignorelog true`
check_return_code "Import solid data result parsing"
rm ngs_import_solid_result.txt


# Import genome
echo ""
echo "Import genome"
$SERVERCMD -A import -f automaticimport -s "$IMPORTPATH/reference.fa" ${destdir} -O import_result.txt
check_return_code "Import genome"
refdata=`$PARSECMD -f import_result.txt -p "--references" -c "reference"`
check_return_code "Import genome result parsing"
rm import_result.txt


# Do readmapping
echo ""
echo "Read Mapping"
$SERVERCMD -A read_mapping ${soliddata} ${destdir} ${refdata} -O read_mapping_result.txt
check_return_code "Read Mapping"
echo $PARSECMD -f read_mapping_result.txt -p "-i" -c "mapping"
readmap=`$PARSECMD -f read_mapping_result.txt -p "-i" -c "Reads"`
check_return_code "Read Mapping result parsing"
rm read_mapping_result.txt

# Quality-based Variant Detection
echo ""
echo "Quality-based Variant Detection"
$SERVERCMD -A quality-based_variant_detection --create-table true --min-coverage 1 --variant-in-forward-revers
check_return_code "Quality-based Variation Detection"
table=`$PARSECMD -f quality_result.txt -p "-s" -c "(Reads)"`
check_return_code "Quality-based Variation Detection result parsing"
rm quality_result.txt

# Export variant table to excel
echo ""
echo "Export variant table to excel"
$SERVERCMD -A export -e excel_2010_exporter -d . ${table} -O table_export_result.txt
check_return_code "Export variant table to excel"
file=`$PARSECMD -f table_export_result.txt -O File`
check_return_code "Export variant table to excel result parsing"
rm table_export_result.txt

# Workflow completed
echo ""
echo "Workflow completed succesfully"
echo "Variant table: ${file}"
```

Chapter 4

Usage for all commands

A complete overview of usage for all commands can be found at http://www.clcsupport.com/clcservercommandlinetools/current/index.php?manual=Usage_all_commands.html