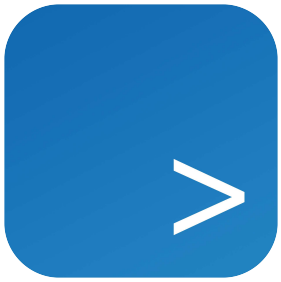


CLC **Server Command Line Tools**

USER MANUAL

Manual for

CLC Server Command Line Tools 23.0.4

Windows, macOS and Linux

May 17, 2023

This software is for research purposes only.

QIAGEN Aarhus
Silkeborgvej 2
Prismet
DK-8000 Aarhus C
Denmark

Contents

Contents

Chapter 1

Introduction

Welcome to the user manual of *CLC Server Command Line Tools 23.0.4*.

The *CLC Server Command Line Tools* provide a command line client for CLC Server solutions[1]. Using this client, tasks can be started on CLC Servers, including bioinformatics analyses, data import and export, and utility data operations such as moving, renaming, and deleting data. Some data and maintenance related tasks can only be carried out by users in the admin group, such as emptying all users' recycling bins or installing plugins.

The *CLC Server Command Line Tools* is particularly well suited for work on production environments. Automation and consistency are well supported through inclusion of *CLC Server Command Line Tools* commands in scripts and the use of standard system tools for scheduling tasks.

CLC Server Command Line Tools commands are non-interactive: all data and parameter settings required are specified up front in the command, making it very quick to launch complex jobs once the desired settings have been determined.

Choosing a *CLC Server* client

We provide two types of clients for the *CLC Server*, the *CLC Server Command Line Tools* and the graphical CLC Workbenches. Here we outline some considerations that may be useful when considering which client type to use for your work.

- For *visualization and interpretation of data* we recommend using a CLC Workbench. If results are generated using the *CLC Server Command Line Tools*, then these can be viewed using a CLC Workbench connected to the same CLC Server the analyses were carried out on. Alternatively, the data can be exported and shared.
- For *explorative work* we recommend using a CLC Workbench. The effects of parameter changes, for example, are easier to interpret using the graphical interface. For many users, selection and management of data is also more intuitive through a graphical interface. 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*.

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

- For *analysis consistency and running analyses in a hands off manner*, either client can be used. With the *CLC Server Command Line Tools*, pipelines of tasks to be run on the *CLC Server can be scripted*. Using a CLC Workbench, pipelines of tasks can be specified in a workflow, which can then be run directly on the *CLC Server* or on the CLC Workbench itself. Workflows can also be installed on a *CLC Server*, and such workflows can be launched using either client type.
- *Automation* is supported by the *CLC Server Command Line Tools*, where standard scheduling tools can be used to schedule the launching of commands or scripts.

Chapter 2

Installation

The *CLC Server Command Line Tools* can be downloaded from https://digitalinsights.qiagen.com/products-overview/discovery-insights-portfolio/enterprise-ngs-solutions/clc-server-command-line-tools/ and is available for Windows, Mac and Linux. You can install the tools on any computer that can connect to your *CLC Server*.

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

- Windows 8, Windows 10, Windows 11, Windows Server 2012, Windows Server 2016, Windows Server 2019 and Windows Server 2022
- Mac: macOS 11, 12, and 13
- Linux: RHEL 7 and later, SUSE Linux Enterprise Server 12 and later. The software is expected to run without problem on other recent Linux systems, but we do not guarantee this.
- 64 bit
- 1 GB RAM required
- 2 GB RAM recommended
- 1024 x 768 display required
- 1600 x 1200 display recommended

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

Chapter 3

Basic usage

Once installed, there will be four 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 7).
- `clcserverkeystore` - a helper tool for enabling passwords to be handled securely (see section 4).
- `clcserversslstore` - a helper tool for managing SSL certificates (see section 5)

Getting help

Run the `clcserver` command with no arguments or with an incomplete set of arguments to see information about what can be run on the server. Below is a list of command forms, and the type of information produced using them.

- `clcserver` Running just this command with no arguments returns general information and options.
- `clcserver -S <server> -U <username> -W <password or token>` A command of this form returns a list of all the commands that can be run on that server.
- `clcserver -S <server> -U <username> -W <password or token> -A <task name>` A command of this form returns all the options available when submitting that task to the server.

The clcserver command - details

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 4 for how to avoid entering passwords in clear text.

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

–A <command to be executed on server>

This list returned includes analysis and utility tools, as well as workflows installed on the CLCServer.

If you supply the -A flag followed by a tool or workflow name, but do not provide additional required options, then a listing of the available options will be returned.

Inputs to tools and workflows, and output destinations, are specified using CLC URLs. Details about these are provided in chapter 6.

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 7 for information about parsing this file. By default, this file is placed in your working directory."

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>

Options available for managing and querying jobs and results include:

–Y Execute the command asynchronously. The returned process ID can be used to query for status and results by using the **–I** and **–R** flags, respectively.

–I <process IDs> Return information about the listed processes. If a list of process IDs is not provided, information about all processes submitted by the current user are returned.

–R <process IDs> Returns the results of the finished processes. Results for a given process can only be retrieved once using this option. To cancel a specific process use **–R <process ID> -R cancel**. To cancel all processes owned by current user, use **–R cancel-all**. Using that option as an administrative user will cancel all processes.

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*.

Hint: When launching multiple tasks via a script, putting `sleep 10` between the commands can help keep the memory needed for the waiting command to a minimum. It can also be worth checking that the number of user processes allowed is sufficient (ulimit setting).

Chapter 4

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
```

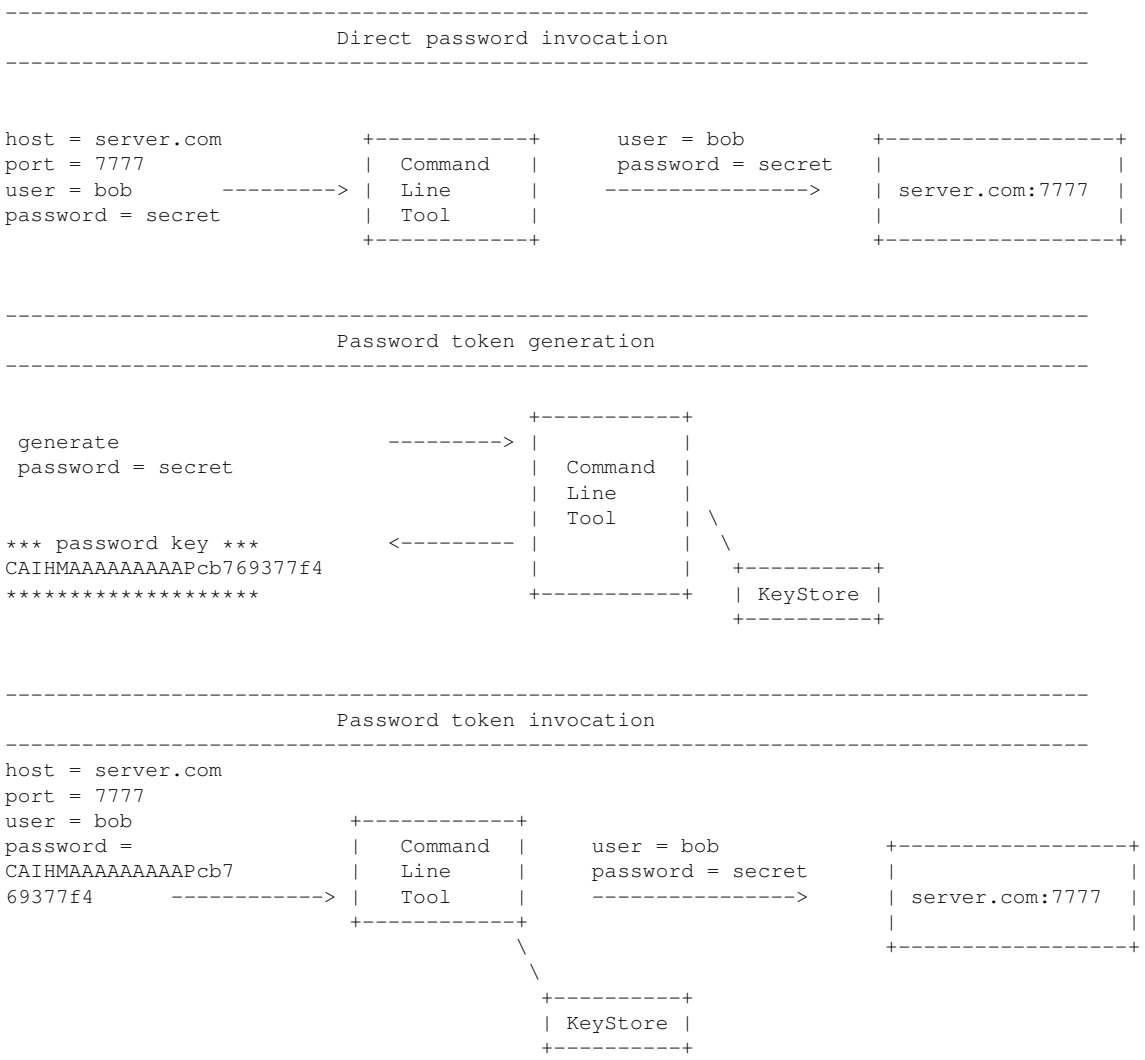
```
clcserver -S server.com -U bob -W CAIHMAAAAAAAAAPcb769377f4
```

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.



Chapter 5

Managing SSL certificates

The `clcserver` command will automatically detect and use SSL if present on the port it connects to. However, if the certificate is untrusted it will refuse to login. In order to connect to a server, its certificate must be added to the trust-store by using the `clcserversslstore` utility.

When invoking `clcserversslstore` it is possible to both list and add new certificates to the trust-store. Certificates are added by providing the program with the connection information (via the -S, -P, -U, and -W parameters):

```
clcserversslstore –S server.com –U bob –W secret –P 7778
```

If the port connected to is indeed an SSL-enabled port, the program will ask if the certificate should be trusted for future `clcserver` invocation:

```
The server (server.com) presented an untrusted certificate with the
following attributes:

SUBJECT
=======
Common Name         : server.com
Alternative Names   : N/A
Organizational Unit: Enterprise
Organization        : CLC Bio
Locality            : Aarhus N.
State               : N/A
Country             : DK

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

```
FINGERPRINTS
============
SHA-1               : A5 F6 8D C4 F6 F3 C2 44
SHA-256             : 49 B5 0B 04 3C 3A A1 E2 D1 BF 87 10

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

Answering `y` to this will record the certificate in the trust-store, and allow subsequent `clcserver` invocation to connect to the server.

It is possible to list the trusted certificates by invoking the `clcserversslstore` program with the -L argument.

Chapter 6

Input data and output destinations

The location of input data and output destinations are specified using CLC URLs. Where a CLC URL is expected as the value for a parameter, the type is specified in the help for relevant parameters. For example, running the following command:

```
clcserver -S <server> -U <username> -W <password or token> -A assemble_sanger_sequences
```

would result in help text being printed to screen that included the following:

```
-i, --input <ClcObjectUrl>
-d, --destination <ClcServerObjectUrl>
```

CLC URL types are described at the top of the help returned when the `clcserver` command is run with an incomplete set of arguments. Information about CLC URLs is also provided in the table below.

For CLC Object URLs, there are 2 forms:

- **Name form** URLs where the path to the file or folder of interest is given relative to the base of a CLC Server file location.
- **ID form** URLs obtained via CLC software. This form benefits from not being affected by changes to the names of data elements or folders, but they are not human interpretable. How to obtain the ID form is described in section 6.1

Working with workflows is described in chapter 8.

CLC URL type	Used for	URL forms)	Details and Examples)

ClcServerObjectUrl
ClcObjectUrl
	Files and folders in CLC Server file locations.	clc://server/
		clc.//host:port
			Name form examples:
			`clc://server/CLC_Server_Loc/path/to/myreads`
			`clc://host01:7777/CLC_Server_Loc/path/to/myreads`
			ID form examples:
			`clc://server/3123-2131uafd-sads/213-sddsa123-5232`
			`clc://host01:7777/3123-2131uafd-sads/213-sddsa123-5232`

ClcFileUrl	Files and folders in CLC Server import/export directories	clc://serverfile/	Full path to file in a CLC Server import/export directory. Example: `clc://serverfile/full/path/to/impexpdir/reads/Fwd4_R1.fastq` If direct data transfer from client systems has been enabled for the CLC Server, a full path to a file on the local file system can be provided.

ClcCloudFileUrl	Files and folders in internet locations	clc://cloudfile/
		or URLs
			Examples:
			`clc://cloudfile/s3://my-bucket/reads/Fwd4_R1.fastq`
			`s3://my-bucket/reads/Fwd4_R1.fastq`
			`https://www.dropbox.com/1u1sxo/Fwd4_R1.fastq`

Table 6.1: CLC URL types

6.1 Getting the ID form of a CLC URL

The ID form of a CLC URL is obtained via CLC software. There are several ways to do this:

1. From the Navigation Area of a *CLC Workbench*: Select the element in the Navigation Area, and copy the CLC URL, for example by using the keyboard shortcut Ctrl-C, or right-clicking and selecting the option "Copy" from the menu that appears. Then paste the URL where it is needed (figure 6.1).

2. Using the CLC Server web 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. Expand the "CLC URL" section. This shows two versions of the CLC URL, one using the name form and one using the ID form.

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.

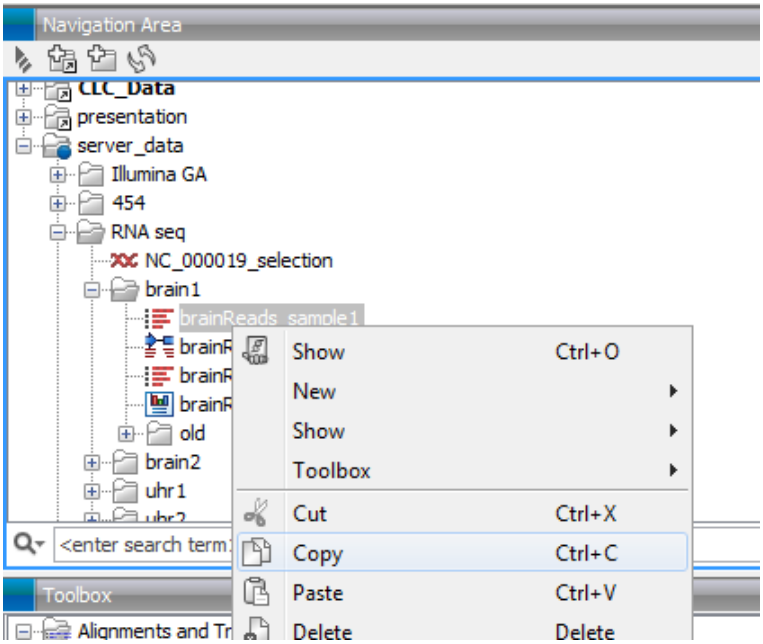


Figure 6.1: *Copying a data element from the Navigation Area of a CLC Workbench copies the CLC URL to the clipboard.*

Chapter 7

Result files and connecting analyses in pipelines

For each run of the `clcserver` command, a summary of the steps taken and the locations of the results, in ID form, are returned to stdout. By adding `-O <filename>` to the `clcserver` command, the locations of the results can also be written to a file.

The typical contents of such a file is shown in the example below. Here, the trim algorithm had been run on a sequence list called `reads`. Three data elements were generated as output, and their names and locations were written to the file specified on the command line using the `-O` option.

```
//
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 Read log
ClcUrl: clc://127.0.0.1:7777/-268177574-CAAAAAAAAAAAAAPc673b0db8c5e724f--5d66a991-12d75090d93--7fff
//
```

When creating pipelines of analyses, you would typically parse such a file for the locations of outputs to be used as inputs for downstream analyses. The `clcresultparser` tool is provided to help with this. This tool searches a file like the one shown above for an expression supplied to it. By default, it returns the locations of data elements where all or part of the Name field matches the search term supplied.

For example, if the file shown above was called `results.txt`, the location of the trimmed reads output could be obtained by running this command:

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

Here, the following would be returned, indicating the location of the data element:

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

The parameters for the `clcresultparser` program are described below. Running the tool without any arguments will also return information about the available parameters.

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

–F <field name to search> Specify the field in the file to search against. (default: Name)

–O <field contents to report> Field to be printed for matching entries. (default: ClcUrl)

–c <text> Text to search for. If nothing is found, the command returns with exit code 1.

–n <text> Entries where this text is found are not reported. (Case insensitive)

–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`).

–p <prefix text> When more than one match is found, the data locations for all matches will be output as a space-separated list. With this option, you stipulate the character(s) to separate the list with. E.g. If you need to send several files output by the `clcresultparser` command as arguments to `-i` options for the next analysis, you could supply `"-i"`as the argument for the `-p` flag.

–e <integer> The number of entries that are **expected** to be returned. If the number of results does not match this number, 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 can check the error code returned by the previous command by typing `echo $?`)

– –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.

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

Chapter 8

Executing workflows

Available workflows are listed when the `clcserver` command is run with the required information for logging in but without the `-A` flag specified. E.g.

```
clcserver -S <server> -U <username> -W <password or token>
```

Unlocked workflow parameters, i.e. those that can be configured, are listed when the `clcserver` command is run with the required information for logging in, and include `-A` flag followed by the workflow name.

```
clcserver -S <server> -U <username> -W <password or token> -A <workflow-name>
```

Workflows must be installed on the *CLC Server* for them to be available to launch using the *CLC Server Command Line Tools*.

Workflows are described in detail here: http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Workflows.html

Running Template Workflows

Template workflows provided by CLC Workbenches, plugins and modules are not installed on the *CLC Server* by default. To launch such workflows using the *CLC Server Command Line Tools*, make a copy of the workflow, save that copy, and then install it on the *CLC Server*. Doing this is described at: https://resources.qiagenbioinformatics.com/manuals/clcserver/current/admin/index.php?manual=Installing_configuring_workflows.html.

Each input is specified using a dedicated parameter call

As with tools, each input to a workflow must be specified with its own parameter. Lists of input files as an argument to a single parameter are not accepted. For example, for 2 input elements, the general form would be:

```
--<parametername>-workflow-input <firstdataelement> --<parametername>-workflow-input <anotherdataelement>
```

On-the-fly import using the CLC Server Command Line Tools

For each workflow Input element, you *either* specify CLC data elements, *or* you specify on-the-fly

import of data. On-the-fly import indicates that data should be imported as the first step taken when the workflow is run.

Configuring inputs in workflow designs, including on-the-fly import, is described in the *CLC Genomics Workbench* at http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Configuring_input_output_elements.html.

Parameters relating to a workflow input are listed when an incomplete command of the following form is run:

```
clcserver -S <servername> -U <username> -W <passwd-or-token> -A <workflowname>
```

For a workflow with a default configuration, these parameters would be:

- `<parameter-name>-input` Used to specify data held in a CLC Server data location or to specify CLC format files in another location, e.g. AWS S3.

  When working with CLC format files in a remote location, e.g. AWS S3, you can provide these as input to workflows by specifying the URLs directly. For example:

  ```
  clcserver <login parameters> -A <workflowname> \\
   --<parameter-name>-input s3://bucketname/path/to/myreads.clc
  ```

- `<parameter-name>-import-command` Used for on-the-fly import. The list of available importers is provided in the command information. One of these is then expected as the argument for this parameter. When using this option, you then also provide parameters relevant to the importer specified.

Only one of these should be specified for a given workflow input.

An importer's parameters are listed when an incomplete command is run that contains the `<parameter-name>-import-command` parameter with an importer name as the argument. For example, to see a list of parameters for use with the Illumina importer, a command of this form could be run:

```
clcserver -S<servername>-U <username>-W <passwd-or-token> -A <workflowname> \\
-<parameter-name>-import-command ngs_import_illumina
```

Each input file needs to be specified individually. E.g. when using the the Illumina importer to import 2 files, part of the command would look like:

```
-<parameter-name>-import-command ngs_import_illumina \\
--workflow-input-select-files clc://serverfile/full/path/to/impexpdir/reads/seq _R1.fastq \\
--workflow-input-select-files  clc://serverfile/full/path/to/impexpdir/reads/seq1_R2.fastq
```

A list of input files is not accepted as the argument for input parameters.

Parameter names in workflows

The name of parameters within workflows are unique. This is ensured 2 ways:

- Each parameter name includes the name of the workflow element it is associated with and every element in a workflow has a unique name.

- If parameter names within a workflow element are identical, numbers are appended to those names for use with the *CLC Server Command Line Tools*. Although this ensures parameter names are unique, we recommend that when editing workflows, duplicate parameter names within a given workflow element are avoided to avoid confusion.

Chapter 9

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 CLC Server, can empty the recycle bin associated with CLC Server file locations. This is because the recycle bin is a shared location for any given CLC 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 recycle 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 token> -A empty_recycle_bin -t clc://server/$LOCATIONNAME
```

Figure 9.1: *How to set up a script that automatically empties the recycle bin.*

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