CLC **Server Command Line Tools**

USER MANUAL

Manual for

CLC Server Command Line Tools 24.0.1

Windows, macOS and Linux

July 1, 2024

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 24.0.1*.

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 10, Windows 11, Windows Server 2016, Windows Server 2019 and Windows Server 2022
- Mac: macOS 12, 13 and 14
- 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>

If you supply the -A flag followed by a tool or workflow name, but do not provide all the required parameters in the command, then a listing of configurable parameters is returned, as well as a list of the missing required parameters.

When optional parameters are not explicitly included in a command, the default values are used by the tool or workflow.

Input data is specified using an input parameter per input element or input file, while a single parameter specifies the folder to save CLC outputs to or the folder an exporter should export files to. The location of input data for an analysis, and the locations to save outputs or export results to, are specified using URLs (see chapter 6).

–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. This is an optional parameter, but useful when working with scripts as 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.

```
-------------------------------------------------------------------------------
                          Direct password invocation
-------------------------------------------------------------------------------


host = server.com           +------------+      user = bob          +------------------+
port = 7777                 |  Command   |      password = secret   |                  |
user = bob       ---------> |  Line      |     ---------------->    | server.com:7777  |
password = secret           |  Tool      |                          |                  |
                            +------------+                          +------------------+


-------------------------------------------------------------------------------
                          Password token generation
-------------------------------------------------------------------------------

                                        +-----------+
 generate                     ---------> |           |
 password = secret                       |  Command  |
                                         |  Line     |
                                         |  Tool     | \
*** password key ***          <--------- |           |  \
CAIHMAAAAAAAAAPcb769377f4                |           |   +----------+
********************                    +-----------+   | KeyStore |
                                                        +----------+


-------------------------------------------------------------------------------
                          Password token invocation
-------------------------------------------------------------------------------
host = server.com
port = 7777
user = bob                  +------------+
password =                  |   Command  |     user = bob            +------------------+
CAIHMAAAAAAAAAPcb7          |   Line     |     password = secret     |                  |
69377f4      ------------> |   Tool     |     ---------------->     | server.com:7777  |
                            +------------+                           |                  |
                                         \                           +------------------+
                                          \
                                           +----------+
                                           | KeyStore |
                                           +----------+
```

Chapter 5

Managing SSL certificates

A secure connection is established to a *CLC Server* by supplying the relevant port number, usually `8443`, to the `clcserver` command. E.g.

```
clcserver -S <servername> -U <username> -W <passcode> -P 8443
```

The `clcserver` command uses SSL if it is present on the port it connects to.

If the server certificate is untrusted, the connection will not be established and login will fail with the message: `SSL Handshake failed. Check certificate.`

To specify that the certificate should be trusted, it must be added to the *CLC Server Command Line Tools* truststore. To do this, run the `clcserversslstore` tool with the relevant connection details, e.g.

```
clcserversslstore -S server.com -U bob -W secret -P 8443
```

The details of the certificate are then displayed, and the option to add the certificate to the truststore (y) or not to do so (n) is provided.

```
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       : QIAGEN
Locality           : Aarhus N
State              : N/A
Country            : DK

ISSUER
=======
Common Name        : server.com
Organizational Unit: Enterprise
```

```
Organization        : QIAGEN
Locality            : Aarhus N
State               : N/A
Country             : DK

FINGERPRINTS
============
SHA-1               : 21 34 6E 8D 9B 01 33 B5 D6 40 73 56 7A 2F 87 A7 EE 3C 21 44
SHA-256             : E5 7F F3 19 8A C1 53 16 00 39 EC F6 65 B3 15 AD 6F 71 DC 2C

VALIDITY PERIOD
===============
Valid From          : 4 Apr 2024
Valid To            : 4 Apr 2025
Trust this certificate? [yn]
```

After the certificate is added to the *CLC Server Command Line Tools* truststore, the `clcserver` tool can be used to connect securely to the *CLC Server*.

Add the `-L` flag to the `clcserversslstore` command, along with the connection information, to list the certificates trusted by the *CLC Server Command Line Tools*.

Chapter 6

Specifying locations of inputs, outputs and exports

The location of input data for an analysis, and the locations to save outputs or export results to, are specified using URLs. 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 a table below.

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>
```

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

Information about providing inputs to workflows being run on the *CLC Server Command Line Tools* are provided in chapter 8.2.

Using data stored on AWS S3 as input

Data on S3 specified as input for an analysis on the *CLC Server* is downloaded to the *CLC Server* at the start of the analysis. Note that AWS charges for data download.

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

Data can be analyzed directly on AWS, with results saved to AWS S3, using *CLC Genomics Cloud* setup. Please refer to the CLC Cloud Module for further information about this: https://resources.qiagenbioinformatics.com/manuals/clccloudmodule/current/index.php?manual=Overview_CLC_Genomics_Cloud.html.

Exporting results

Data can be exported to compatible formats using an export command or using workflows containing export elements. The location to save the exported file to is specified using a ClcFileURL, ClcCloudFileUrl or AWS S3 URL.

Note that when an export is run on the *CLC Server Command Line Tools* and exported to AWS S3, the exported file will be placed in a subfolder of the specified location with a name made up of a combination of the export file name and a unique string. This is to avoid the danger of overwriting existing files.

Further information about exporting results from workflows is in chapter 8.3.

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.

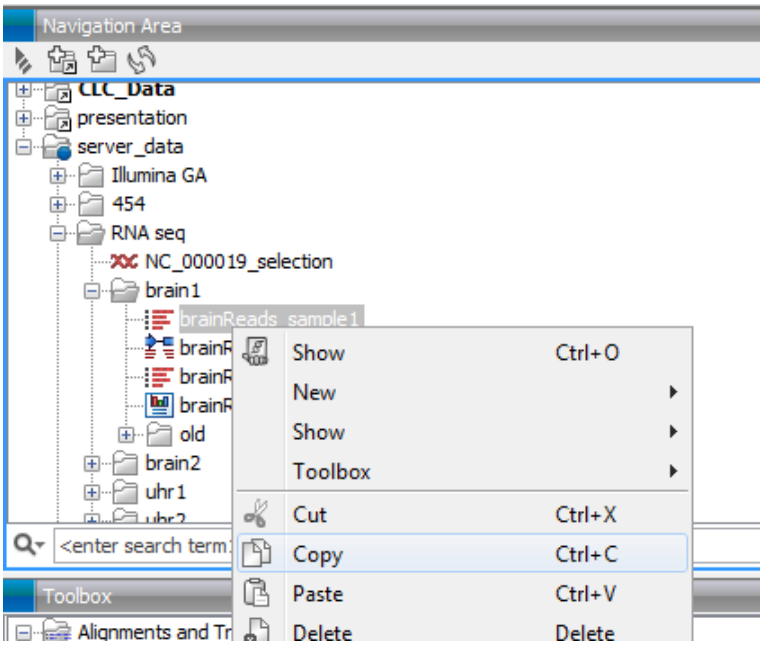

Figure 6.1: *Copying a data element from the Navigation Area of a CLC Workbench copies the CLC URL to the clipboard.*

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.

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

Launching workflows

Workflows consist of sets of connected tools, with outputs from one analysis stage being used as input to later stages. Workflows are easily created using the graphical workflow editor in a *CLC Workbench*, and once installed on a *CLC Server*, they can be launched using a single command. Template workflows for many common analysis pipelines are delivered with the *CLC Genomics Workbench* and with several of its plugins and modules. Copies of template workflows can be installed on the *CLC Server*, like any other workflow.

By default, workflows installed on the *CLC Server* can be executed by all users, but access to each workflow can be controlled by the administrator.

The sections in this chapter focus on launching workflows using the command line:

- General information about launching workflows: section 8.1
- Providing input data: section 8.2
- Saving outputs and exporting results: section 8.3
- Launching workflows containing Iterate elements: section 8.4

General information about workflows, including creating and configuring them, is at `http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Workflows.html`.

Information about installing and configuring workflows on a *CLC Server* is at `https://resources.qiagenbioinformatics.com/manuals/clcserver/current/admin/index.php?manual=Installing_configuring_workflows.html`.

8.1 General workflow launching information

Listing available workflows

Workflows installed on the *CLC Server* 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>
```

Configuring parameter values in a workflow

To see the configurable parameters for a given workflow, run the above command with the workflow name after `-A`, i.e. a command of the form:

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

Workflow authors decide which parameters are configurable when launching a workflow, and which are not. Non-configurable parameters are those that have been locked in the workflow design. Configurable parameters are not locked. For further details, see https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Basic_configuration_workflow_elements.html.

Parameter name uniqueness in workflows

The name of parameters within workflows are unique. This is ensured in 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 duplicate parameter names within a given workflow element are avoided to minimize confusion. Editing parameter names is described at https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Basic_configuration_workflow_elements.html

8.2 Providing input data for analyses on a CLC Server

All input data must be in a location accessible to the *CLC Server*[1]. Input data includes the data being analyzed, and in many cases, reference data.

Parameter names for providing inputs reflect the name of the Input element in the workflow design, taking the form `<input-element-name>-<parameter-name>`.

CLC format data in CLC Server File System Locations or in remote locations accessible via http, https, or S3 URL can be provided as inputs to workflows. Data in other formats or in other locations can be imported at the start of the workflow run using on-the-fly import. These options are described in detail below.

Data in remote locations specified as input for an analysis on the *CLC Server* is downloaded at the start of the workflow run. Note that AWS charges for data download.

Data can be analyzed directly on AWS, with results saved to AWS S3 using a *CLC Genomics Cloud* setup, as described in the *CLC Cloud Module* manual: https://resources.qiagenbioinformatics.com/manuals/clccloudmodule/current/index.php?manual=Overview_CLC_Genomics_Cloud.html.

[1]The information in this section relates to launching analyses to run on the *CLC Server*. Please refer to the *CLC Cloud Module* manual for information relevant to submitting analyses to run on a *CLC Genomics Cloud* setup.

Providing CLC data as input to workflows

The expected value for each parameter specifying CLC format data to use as input is a ClcServerObjectUrl or an http, https, or S3 URL. Parameter names for providing CLC format data take the form: `<input-element-name>-workflow-input>`.

Using on-the-fly import

Data stored somewhere other than a CLC Server File System Location can be supplied as input to analyses by using on-the-fly import. For example, using on-the-fly import, FASTQ sequence files would be imported as the first step in the workflow, avoiding the need for running a specific import command before running the workflow.

On-the-fly import can be used for a particular input if, when a partial workflow command is submitted, the listing of available parameters includes one with a name of the form `<input element name>-import-command`. The value for that parameter is the importer to use. When on-the-fly import is used, parameters are also needed to specify the location of the files to be imported. One parameter-value pair is needed for each file. The file location is specified using a ClcFileUrl, a ClcCloudFileUrl or an http, https, or S3 URL. An example of using on-the-fly import for import of Illumina data is provided below.

Many importers require additional configuration information. These options parameter are listed when the incomplete `clcserver` command includes the `<input element name>-<import-command>` parameter. The settings for individual importers are described in the import chapter of the *CLC Genomics Workbench* manual: https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Import_export_data_graphics.html.

On-the-fly import is usually possible when an Input element has been connected to the relevant input channel in the workflow design. However, a workflow author can configure Input elements so on-the-fly import is not allowed (see https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Configuring_input_output_elements.html).

Specifying input data - an example

The parameters in this section relate to the workflow shown in figure 8.1. There are 2 Input elements in this workflow, Sample Reads and Reference Genome.

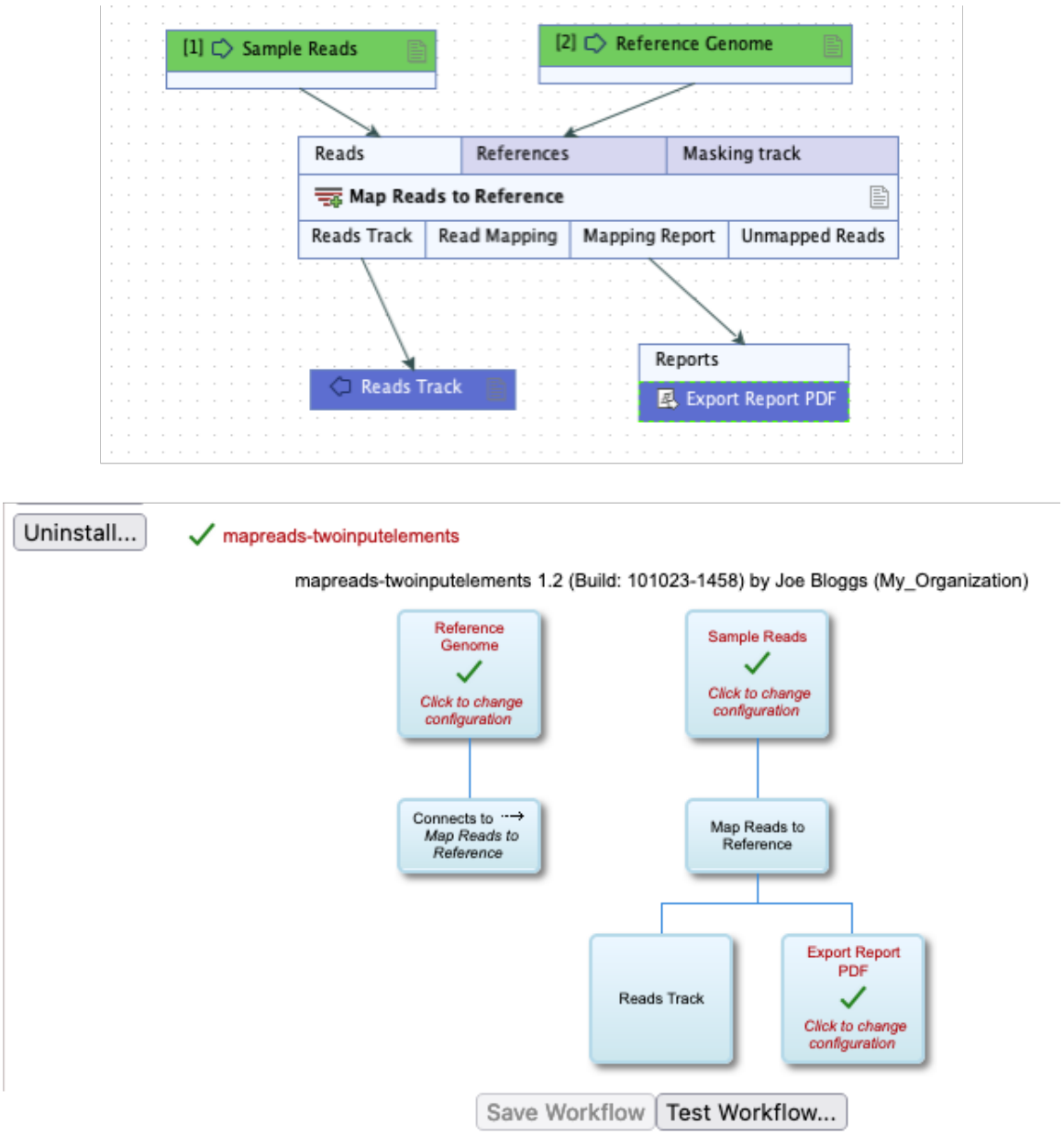

Figure 8.1: *A simple workflow seen in the Workflow Editor of a CLC Workbench (top) and the CLC Server web interface (bottom). Two Input elements are present, Sample Reads and Reference Genome.*

Using this workflow, a command of the following form:

```
clcserver -S <server> -U <username> -W <password or token> -A wf-mapreads-twoinputelements
```

would return the following parameter information relating to inputs:

```
--reference-genome-import-command <      The importer to use for on-the-fly import of input data.
[clc_import, ngs_import_fasta,
ngs_import_genereader,
```

```
ngs_import_illumina,
ngs_import_iontorrent,
ngs_import_mgi_bgi,
ngs_import_pacbio, ngs_import_sanger,
trace_files_import]>
--reference-genome-workflow-input        Workflow Input
<ClcObjectUrl>
--sample-reads-import-command <          The importer to use for on-the-fly import of input data.
[clc_import, ngs_import_fasta,
ngs_import_genereader,
ngs_import_illumina,
ngs_import_iontorrent,
ngs_import_mgi_bgi,
ngs_import_pacbio, ngs_import_sanger,
trace_files_import]>
--sample-reads-workflow-input            Workflow Input
<ClcObjectUrl>
```

For the Sample Reads input element, the parameters available are:

- `--sample-reads-workflow-input` Used to specify CLC format data in a CLC Server File System Location, with a CLC Object URL as the value, or in a remote location, with an http, https, or S3 URL as the value.One parameter-value pair is needed for each input element.
- `--sample-reads-import-command` Used to indicate that the data should be imported as the first step of the workflow (on-the-fly import). One of the available importers is expected as the value. Corresponding `--sample-reads-select-files` parameters, each with a ClcFileUrl as the value, are then used to specify each file to import. One parameter-value pair is needed for each input file.

All data for a given workflow input (e.g. Sample Reads) must *either* be specified using `--sample-reads-workflow-input` parameters *or* it must be imported using a `--sample-reads-import-command` parameter and `--sample-reads-select-files` parameters.

A common situation is to use on-the-fly import for sample data and to use already imported data for reference data. Doing this for the example workflow, the input related parameters could look like:

```
--reference-genome-workflow-input  clc://server/<path-to>/<reference-data-element>  \
--sample-reads-import-command ngs_import_illumina  --reads-paired-reads false  \
--sample-reads-select-file clc://serverfile/<path-to>/SRR6954665_R1.fastq  \
--sample-reads-select-file clc://serverfile//<path-to>/SRR6954668_R1.fastq \
--sample-reads-select-file clc://serverfile//<path-to>/SRR6954672_R1.fastq  \
 --sample-reads-select-file clc://serverfile//<path-to>/SRR6954680_R1.fastq  \
 --sample-reads-select-file clc://serverfile//<path-to>/SRR6954681_R1.fastq  \
 --sample-reads-select-file clc://serverfile//<path-to>/SRR6954682_R1.fastq
```

Obtaining QIAGEN reference data

Many template workflows, and copies of such workflows, refer to reference data supplied by QIAGEN. A *CLC Genomics Workbench* is needed to download these data elements. See https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=QIAGEN_Sets.html for information on how to do this. This data must be downloaded to the *CLC Server*, which must have a CLC_References location configured. For further details, see https://resources.

qiagenbioinformatics.com/manuals/clcserver/current/admin/index.php?manual=Reference_data_management.html.

8.3 Saving workflow outputs and exporting results

Outputs generated by workflows can be saved in CLC Server File System Locations, where they can be viewed and analyzed further, or results can be exported and saved to any place accessible to the *CLC Server*, including AWS S3 locations if an AWS Connection is configured on the server that allows this[2].

Outputs, which are saved in CLC format, are saved using Output elements in a workflow design, while data is exported using an Export element. In figure 8.2, the Reads Track output channel is connected to an Output element, so this output will be saved in CLC format. The Mapping Report output channel is connected to an Export element, so the mapping report will be exported, in this case to a PDF format file.

Because workflow command parameters include the name of the workflow element they are associated with, renaming Export elements in workflow designs can be helpful, especially when a particular exporter is present more than once in a workflow.

[2]The information in this section relates to launching analyses to run on the *CLC Server*. Please refer to the *CLC Cloud Module* manual for information relevant to submitting analyses to run on a *CLC Genomics Cloud* setup.

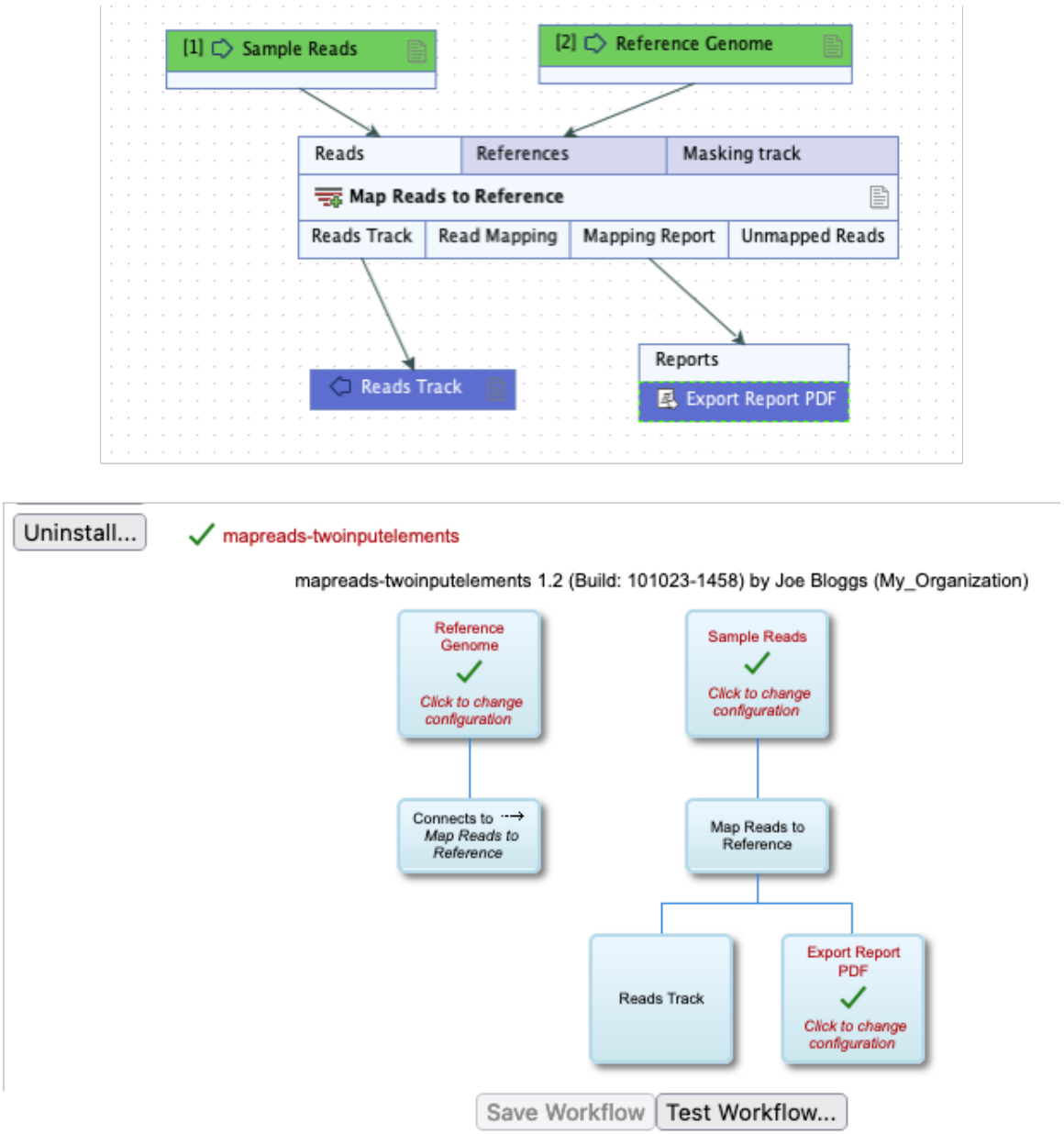

Figure 8.2: *A simple workflow seen in the Workflow Editor of a CLC Workbench (top) and the CLC Server web interface (bottom). An Output element is connected to the Reads Track output channel. An Export element is connected to the Mapping Report output channel.*

Defining where workflow outputs are saved

All outputs are saved under the folder specified using the `-d` or `--destination` parameter. The value expected for analyses run on a *CLC Server Command Line Tools* is a ClcServerObjectUrl (see chapter 6).

In most cases, if the destination is not specified, the analysis outputs are saved to the same folder that the first input element used is in. Exceptions to this are workflows containing export elements, or when one or more set of inputs is not in a CLC Server File Location. In these cases,

the `-d` or `--destination` parameter must be specified.

Defining where exported files are saved

Each exporter has parameters specific to it, including a parameter specifying where the exported file should be saved, which takes the form `--<export-element-name>-export-destination`. The expected value is a URL for the destination folder.

For example, the export location could be a folder in a *CLC Server* Import/export directory, specified using a ClcFileUrl:

```
--<export-element-name>-export-destination clc://serverfile/<path-to>/<destination-folder>/
```

or it could be a remote location accessible from the *CLC Server* that can be specified using a ClcCloudFileUrl, or an http, https, or S3 URL. E.g. using an S3 URL, it could look like:

```
--<export-element-name>-export-destination s3://<bucket-name>/<path-to>/<destination-folder>/
```

When exporting to AWS S3 from the *CLC Server Command Line Tools*, the exported file will be placed in a subfolder of the specified location with a name made up of a combination of the export file name and a unique string. This is to avoid the danger of overwriting existing results, if the same folder is specified for multiple exported files with the same name.

See chapter 6 for more on specifying locations using URLs.

Some exporters require additional information to be provided. These options are described in the relevant importer section of the *CLC Genomics Workbench* manual: https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Data_export.html.

8.4 Launching workflows with Iterate elements

The section focuses on launching workflows that contain Iterate elements using the *CLC Server Command Line Tools*. Iterate elements are a type of control flow element, controlling the flow of data through an analysis. Iterate elements are placed at the top of a branch of a workflow that should be run multiple times, using different inputs in each run. The sets of data to use in each run are referred to as "batch units".

Collect and Distribute elements are, optionally, placed downstream of an Iterate element, where they collect outputs from the upstream iteration block and distribute them as inputs to downstream analyses. Most Collect and Distribute elements have a single input channel and a single output channel and do not require any parameters to be specified on the command line. Writing commands for other situations is described at the end of this section.

General information about control flow elements is provided at https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Control_flow_elements.html.

The steps between an Iterate element and a Collect and Distribute element are referred to as an "iteration block". The workflow in figure 8.3 contains a single iteration block (shaded in turquoise), where steps within that block are run once per batch unit. The Collect and Distribute element collects all the results from the iteration block and sends it as input to the next stage of the analysis (shaded in purple).

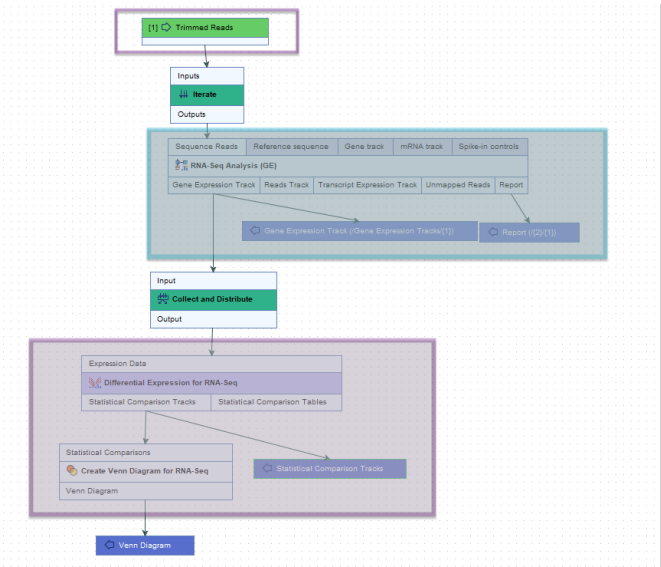

Figure 8.3: *The roles of the Iterate and Collect and Distribute control flow elements are highlighted in the context of RNA-Seq and differential expression analyses. RNA-Seq Analysis lies downstream of an Iterate element, within an iteration block (shaded in turquoise). It will thus be run once per batch unit. Differential Expression for RNA-Seq lies immediately downstream of a Collect and Distribute element, and is sent all the expression results from the iteration block as input for a single analysis.*

The following are key to launching workflows containing an Iterate element:

- Specifying how batch units are defined: by the organization of the input data or using metadata.
- For batch units specified using metadata:
  - Indicating whether the metadata in a CLC Metadata Table or in an external file, specifically an Excel, CSV or TSV format file, and
  - Specifying the metadata column defining the grouping of the data.

For workflows with a single Iterate element that has a single input channel and a single output channel, and where the batch units are based on the organization of the input data, no parameters relating to the Iterate element need to be provided in the command. In other cases, the parameters below need to be specified. Parameter names start with the workflow element name, which in this case was the default name, `Iterate`.

- `--iterate-iterate-units` Used to specify how the batch units are defined:
  - `SIMPLE` (default) Based on the organization of the input data.
  - `METADATA` Based on metadata that will be provided.

    Note that when launching a workflow containing a tool requiring metadata as input, for example **Differential Expression for RNA-Seq**, batch units *must* be specified using metadata. The metadata provided to define the batch units is also used in the analysis step(s) requiring metadata.

- `--iterate-metadata-sources` Required when batch units are defined using metadata to specify how the metadata will be provided.
  - `TABLE_OBJECT` In a CLC Metadata Table
  - `FILE` In an Excel, CSV or TSV file
- `---iterate-metadata-table` When `TABLE_OBJECT` is defined as the metadata source, specify this parameter and provide a CLC Object URL for the CLC Metadata Table to use as the value.
- `--iterate-metadata-file` When `FILE` is defined as the metadata source, specify this parameter and provide the location of the Excel, TSV or CSV format file to use as the value.
- `--iterate-metadata-table-columns` The metadata column containing the information for grouping the data into batch units. One parameter-value pair is expected per input channel in the Iterate element.

In cases where the Iterate element has multiple input channels, the first input channel is considered the primary input channel by default. To specify a different primary input channel, use the `--iterate-primary-input-channel` parameter. An integer value is expected, where the first channel is specified with the value 0, the second channel is specified with the value 1, and so on.

Further information about defining batch units is provided at https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Running_workflows_in_batch_mode.html.

Launching workflows containing Collect and Distribute elements

The most common situation is to have a Collect and Distribute element with a single input channel and a single output channel, as is the case in the example in figure 8.3. With this design, the results from all the batch units in the upstream iteration block are collected and passed on together as input to the connected downstream step(s). Such Collect and Distribute elements do not require any parameters to be defined on the command line.

Where the Collect and Distribute element has more than one output channel, the parameters below must be specified. Parameter names start with the workflow element name, which in this case was the default name, `Collect and Distribute`.

- `--collect-and-distribute-group-by-metadata-column` Provide the name of the metadata column to use to group the data, where each group is sent individually as input to the connected downstream analysis step(s). The values in this column are mapped to the relevant Collect and Distribute output channel using the parameter below.
- `--collect-and-distribute-output-mapping` Define the Collect and Distribute output channel that data should flow through by mapping each value in the relevant metadata column, specified using the parameter above, to an output channel. The format required is `'<sample-information>=<output channel name>'`. For example, if samples with a metadata value "Treated" should flow through an output channel called "Type 1", the value would be `'Treated=Type 1'`. One parameter-value pair is needed for each mapping.

Template workflow example using Iterate and Collect and Distribute elements

The **RNA-Seq and Differential Gene Expression Analysis** template workflow, distributed with the *CLC Genomics Workbench*, provides an example of using Iterate and Collect and Distribute elements. It is described in detail at https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=RNA_Seq_Differential_Gene_Expression_Analysis_workflow.html.

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.