eccenca
command your data!

Corporate Memory Control (cmemc)

v20.06.1

System Manual

Contents

1 Introduction

This manual describes how to install and set up *eccenca Corporate Memory Control* (cmemc), a command line client for eccenca Corporate Memory. cmemc is intended for system administrators and Linked Data Expert, who wants to automate / remote control activities on Corporate Memory.

To use this manual, cmemc users should have basic knowledge on command line interfaces, terminal usage and config file creation and editing.

This system manual includes the following parts:

- *Installation*
- *Configuration*

This document covers installation and basic usage pattern of cmemc and is not intended to be complete in terms of being a reference for all available options and commands. However, cmemc provides detailed documentation for users via the `--help` option.

1.1 About eccenca Corporate Memory Control (cmemc)

cmemc is the eccenca Corporate Memory Command Line Interface (CLI). It is developed in python and build and delivered as a stand alone single binary for Linux and Windows.

Main features of cmemc include:

- List, import, export, delete and open graphs.
- List, import, export, create and delete projects.
- List and execute local as well as remote SPARQL queries.
- List, execute, open or inspect workflows.
- Import and export the workspace.

1.2 Scope of delivery

The cmemc release package consists of the following files:

- `cmemc` - the Linux ELF 64-bit LSB executable (x86-64, version 1 (SYSV), dynamically linked), tested with Ubuntu
- `cmemc-rhel` - the Linux ELF 64-bit LSB executable (x86-64, version 1 (SYSV), dynamically linked), tested with RHEL
- `cmemc-macosx` - the Mach-O 64-bit executable (x86_64), tested with macOS Mojave

- `cmemc.exe` - the PE32+ executable (console, x86-64), for Microsoft Windows, tested with Windows 10
- `cmemc_vXX.YY_SystemManual.pdf` - this document

2 Installation

Since cmemc is a stand alone binary, installation is not needed. cmemc can be started from a local path and also from a central binary path, such as `/usr/local/bin`. The only needed installation activity is to copy the binary to a path on your system which is in the `PATH` variable.

2.1 Linux installation

As a first step, output the content of the `PATH` variable in order to determine, the path to use:

```
user@ubuntu:/home/user/$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Then unzip the distribution package and copy the binary to an acceptable path (in this example: `/usr/local/bin`)

Note: XXX depends on your actual version

```
user@ubuntu:/home/user/$ unzip cmemc-vXXX.zip
user@ubuntu:/home/user/$ cp cmemc-vXXX/cmemc /usr/local/bin
user@ubuntu:/home/user/$ chmod +x /usr/local/bin/cmemc
```

Finally, test your installation.

Note: XXX depends on your actual version

```
user@ubuntu:/home/user/$ cmemc --version
cmemc, version XXX
```

In case you are using bash or zsh as your shell, you can optionally enable tab completion for cmemc. This is documented on the click framework homepage[1].

In order to enable tab completion with **bash** run the following command in your shell:

```
eval "$(_CMEMC_COMPLETE=source cmemc)"
```

In order to enable tab completion with **zsh** run the following command in your shell:

[1] https://click.palletsprojects.com/en/7.x/bashcomplete/#activation

```
eval "$(_CMEMC_COMPLETE=source_zsh cmemc)"
```

You may want to add the corresponding line to your `.bashrc` or `.zshrc` file for your convenience.

2.2 MacOS installation

Installation on MacOS is similar to installation on Linux. Please copy the unpacked `cmemc-macosx` binary to a directory in your PATH (renaming it to `cmemc`). Since **zsh** is the default shell on MacOS, use the zsh completion as described above.

2.3 Windows installation

The installation for Windows is similar. As a first step, unzip the distribution file (`cmemc-vXXX.zip`), then open `cmd.exe` and go to the directory where the cmemc files are extracted. Then you can start `cmemc.exe`.

Note: XXX depends on your actual version

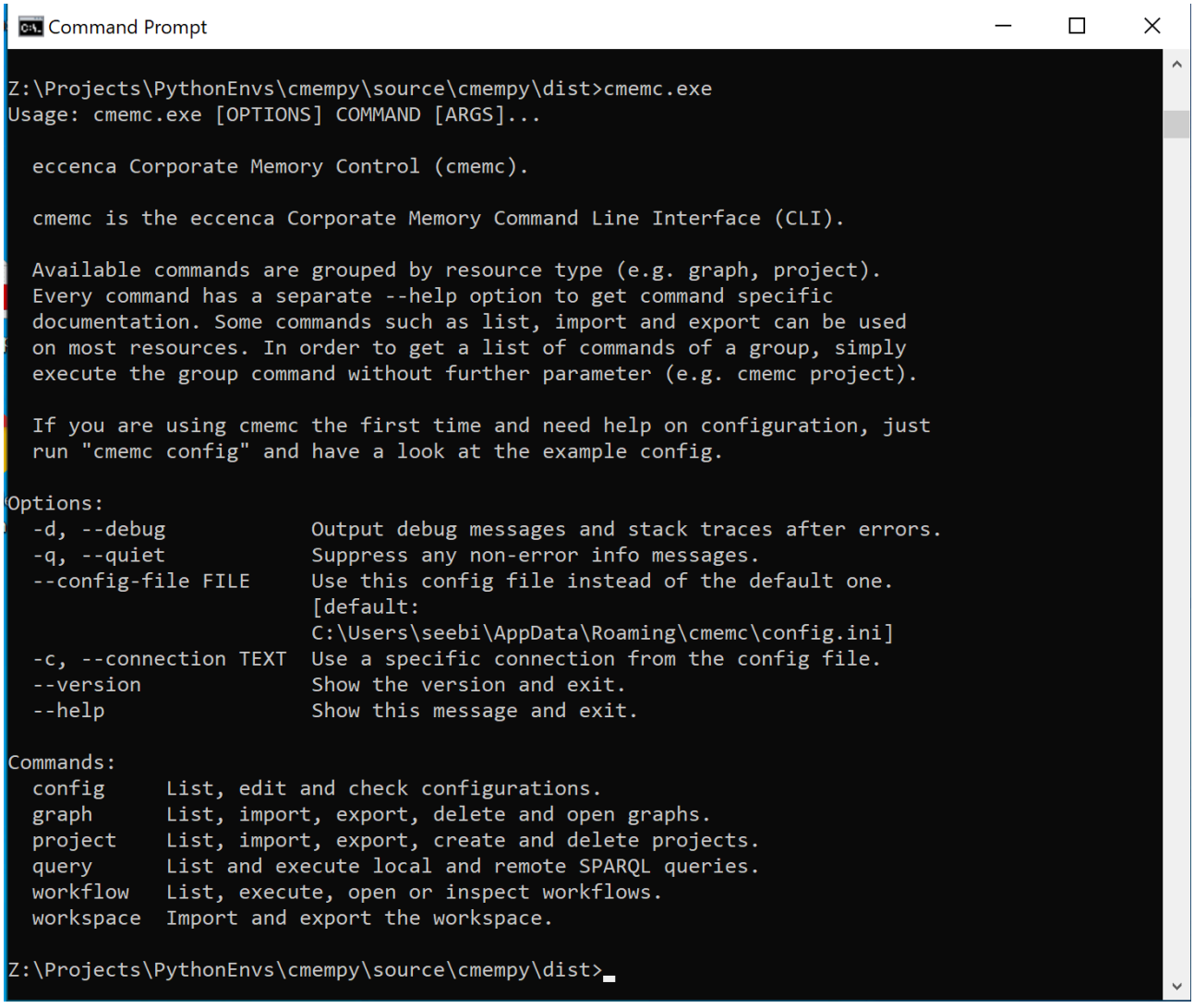

Figure 2.1: Example execution of cmemc under Windows

3 Configuration

cmemc needs to know where your Corporate Memory is deployed. For this, you need to provide some key variables in a configuration file. Per default, cmemc looks for this configuration file on a reasonable place depending on your operating system.

For Linux, this is `$HOME/.config/cmemc/config.ini`.

For Windows, this is `%APPDATA%\cmemc\config.ini`

Note: `USER` is your actual user name.

Once you start cmemc the first time, it will create an empty config file at this location and will output a general introduction. In order to do so, open the terminal application of your choice.

Note: All further examples given here are based on Linux commands. For Windows, the output is the same, however, you need to start cmemc as `cmemc.exe`.

```
user@ubuntu:/home/user/$ cmemc
Empty config created: /home/user/.config/cmemc/config.ini
Usage: cmemc [OPTIONS] COMMAND [ARGS]...

  eccenca Corporate Memory Control (cmemc).

  cmemc is the eccenca Corporate Memory Command Line Interface (CLI).

  Available commands are grouped by affecting resource type (such as graph,
  project and query). Each command / group has a separate --help screen to
  get command / group specific documentation.

  In order to see possible commands in a command group, simply execute the
  group command without further parameter (e.g. cmemc project). Some
  commands such as list, import and export can be executed in most command
  groups. Please have a look at the cmemc manual page for more
  information:

                      https://eccenca.com/go/cmemc

Options:
  -d, --debug            Output debug messages and stack traces after errors.
```

```
  -q, --quiet            Suppress any non-error info messages.
  --config-file FILE     Use this config file instead of the default one.
                         [default: /Users/seebi/Library/Application
                         Support/cmemc/config.ini]
  -c, --connection TEXT  Use a specific connection from the config file.
  --version              Show the version and exit.
  -h, --help             Show this message and exit.

Commands:
  config     List, edit and check configurations.
  graph      List, import, export, delete and open graphs.
  project    List, import, export, create and delete projects.
  query      List and execute local and remote SPARQL queries.
  workflow   List, execute, open or inspect workflows.
  workspace  Import and export the workspace.
```

You should edit now edit your config file and add credentials and URL parameter to your Corporate Memory deployment. You either search the config manually in your home directory or you can use the `config edit` command, which opens the config file in your default text editor.

```
user@ubuntu:/home/user/$ cmemc config edit
Open editor for config file /home/user/.config/cmemc/config.ini
```

The rules for the config file are similar to a Windows INI file and are explained in detail at docs.python.org[1]. Here is an basic example:

```
[my-local]
CMEM_BASE_URI=http://localhost/
OAUTH_GRANT_TYPE=client_credentials
OAUTH_CLIENT_ID=cmem-service-account
OAUTH_CLIENT_SECRET=c9c12831-000c-464b-9b1d-2d8b7e20df6a
```

This basically creates a named section `my-local` which is a connection to a Corporate Memory deployment on `http://localhost`. The authorization will be done with a system account `cmem-service-account` and the given client secret. Using this combination of config parameter is based on a typical installation where all components are available under the some hostname.

However, if you need to fine tune all locations, the following config file parameter can be used in addition to this example:

- `DI_API_ENDPOINT` - Data Integration API endpoint, default: `CMEM_BASE_URI/dataintegration`
- `DP_API_ENDPOINT` - Data Platform API endpoint, default: `CMEM_BASE_URI/dataplatform`

[1] https://docs.python.org/3/library/configparser.html

- `OAUTH_TOKEN_URI` - OAuth 2.0 Token endpoint, default: `CMEM_BASE_URI/auth/realms/cmem/protocol/openid-connect/token`
- `OAUTH_GRANT_TYPE` - OAuth 2.0 grant type, default: `client_credentials`
- `OAUTH_USER` - Username to retrieve the token, default: `admin`
- `OAUTH_PASSWORD` - Password to retrieve the token, default: `admin`
- `OAUTH_CLIENT_ID` - OAuth 2.0 client id, default: `cmem-service-account`
- `OAUTH_CLIENT_SECRET` - OAuth 2.0 client secret, default: `secret`
- `SSL_VERIFY` - Use SSL verification for requests to DP/DI default: `True`

In order to verify your configuration, you should try to get a list of graphs via cmemc:

```
user@ubuntu:/home/user/$ cmemc -c my-local graph list
https://vocab.eccenca.com/dsm/
https://ns.eccenca.com/example/data/dataset/
https://vocab.eccenca.com/sketch/
https://ns.eccenca.com/example/data/vocabs/
https://vocab.eccenca.com/shacl/
urn:elds-backend-access-conditions-graph
https://ns.eccenca.com/data/queries/
http://di.eccenca.com/project/cmem
```

If you get a similar list of graphs, you successfully configured cmemc to access your deployment.

4 Reference

This section lists the help texts of all commands as a reference and to search for it.

4.1 Command group: config

```
Usage: cmemc config [OPTIONS] COMMAND [ARGS]...

  List, edit and check configurations.

  Configurations are identified by the section identifier in the config
  file. Each configuration represent a Corporate Memory deployment with its
  specific access method as well as credentials.

  A minimal configuration which uses client credentials has the following
  entries:

  [example.org]
  CMEM_BASE_URI=https://cmem.example.org/
  OAUTH_GRANT_TYPE=client_credentials
  OAUTH_CLIENT_ID=cmem-service-account
  OAUTH_CLIENT_SECRET=my-secret-account-pass

  In addition to that, the following config parameters can be used as well:

  SSL_VERIFY=False    - for ignoring certificate issues (not recommended)
  DP_API_ENDPOINT=URL - to point to a non-standad DataPlatform location
  DI_API_ENDPOINT=URL - to point to a non-standad DataIntegration location
  OAUTH_TOKEN_URI=URL - to point to an external IdentityProvider location
  OAUTH_USER=username - username (only if OAUTH_GRANT_TYPE=password)
  OAUTH_PASSWORD=pass - password (only if OAUTH_GRANT_TYPE=password)

  In order to request passwords on start, you can use the following
  parameter instead the PASSWORD parameter: OAUTH_PASSWORD_ENTRY=True
  OAUTH_CLIENT_SECRET_ENTRY=True.

Options:
  -h, --help  Show this message and exit.
```

```
Commands:
  check  Check the status of deployment.
  edit   Edit the user-scope configuration file.
  list   List configured CMEM connections.
```

4.1.1 Command: config check

```
Usage: cmemc config check [OPTIONS] [CONFIGS]...

  Check the status of deployment.

Options:
  -a, --all   Export all (readable) graphs
  -h, --help  Show this message and exit.
```

4.1.2 Command: config edit

```
Usage: cmemc config edit [OPTIONS]

  Edit the user-scope configuration file.

Options:
  -h, --help  Show this message and exit.
```

4.1.3 Command: config list

```
Usage: cmemc config list [OPTIONS]

  List configured CMEM connections.

Options:
  -h, --help  Show this message and exit.
```

4.2 Command group: graph

```
Usage: cmemc graph [OPTIONS] COMMAND [ARGS]...

  List, import, export, delete or open graphs.
```

```
  Graphs are identified by an IRI. The get a list of existing graphs,
  execute the list command or use tab-completion.

Options:
  -h, --help  Show this message and exit.

Commands:
  count   Count triples in graph(s).
  delete  Delete graph(s) from the store.
  export  Export graph(s) as NTriples to stdout (-), file or directory.
  import  Import graph(s) to the store.
  list    List accessible graphs.
  open    Open / explore a graph in the browser.
```

4.2.1 Command: graph count

```
Usage: cmemc graph count [OPTIONS] [IRIS]...

  Count triples in graph(s).

  This command lists graphs with their triple count. Counts are done without
  following imported graphs.

Options:
  -a, --all        Count all graphs
  -s, --summarize  Display only a sum of all counted graphs together
  -h, --help       Show this message and exit.
```

4.2.2 Command: graph delete

```
Usage: cmemc graph delete [OPTIONS] [IRIS]...

  Delete graph(s) from the store.

Options:
  -a, --all   Drop all (writeable) graphs
  -h, --help  Show this message and exit.
```

4.2.3 Command: graph export

```
Usage: cmemc graph export [OPTIONS] [IRIS]...

  Export graph(s) as NTriples to stdout (-), file or directory.

  In case of file export, data from all selected graphs will be concatenated
  in one file. In case of directory export, .graph and .nt files will be
  created for each graph.

Options:
  -a, --all                     Export all (readable) graphs
  --output-dir DIRECTORY        Export to this directory
  --output-file FILE            Export to this file  [default: -]
  -t, --filename-template TEXT  Template for the export file name(s). Used
                                together with --output-dir. Possible
                                placeholders are (Jinja2): {{hash}} (the
                                sha256 hash of the graph URI){{connection}}
                                (from the --connection option) and {{date}}
                                (the current date as YYYY-MM-DD). The file
                                suffix will be appended. Needed directories
                                will be created.  [default: {{hash}}]

  -h, --help                    Show this message and exit.
```

4.2.4 Command: graph import

```
Usage: cmemc graph import [OPTIONS] INPUT_PATH [IRI]

  Import graph(s) to the store.

  If input is an directory, it scans for file-pairs such as xxx.ttl and
  xxx.ttl.graph where xxx.ttl is the actual triples file and xxx.ttl.graph
  contains the graph IRI as one string: "https://mygraph.de/xxx/". If input
  is a file, content will be uploaded to IRI. If --replace is set, the data
  will be overwritten, if not, it will be added.

Options:
  --replace   Replace (overwrite) original graph data.
  -h, --help  Show this message and exit.
```

4.2.5 Command: graph list

```
Usage: cmemc graph list [OPTIONS]

  List accessible graphs.

Options:
  --raw                          Outputs raw JSON.
  --filter [readonly|writeable]  Filter list based on access conditions.
  -h, --help                     Show this message and exit.
```

4.2.6 Command: graph open

```
Usage: cmemc graph open [OPTIONS] IRI

  Open / explore a graph in the browser.

Options:
  -h, --help  Show this message and exit.
```

4.3 Command group: project

```
Usage: cmemc project [OPTIONS] COMMAND [ARGS]...

  List, import, export, create or delete projects.

  Projects are identified by an PROJECTID. The get a list of existing
  projects, execute the list command or use tab-completion.

Options:
  -h, --help  Show this message and exit.

Commands:
  create  Create empty new project(s).
  delete  Delete project(s).
  export  Export project(s) to file(s).
  import  Import a project from a file.
  list    List available projects.
```

4.3.1 Command: project create

```
Usage: cmemc project create [OPTIONS] PROJECT_IDS...

  Create empty new project(s).

  This creates one or more new projects. Existing projects will not be
  overwritten.

  Example: cmemc project create my_project

  Projects can be listed by using the 'cmemc project list' command.

Options:
  -h, --help  Show this message and exit.
```

4.3.2 Command: project delete

```
Usage: cmemc project delete [OPTIONS] [PROJECT_IDS]...

  Delete project(s).

  This deletes existing data integration projects from Corporate Memory.
  Projects will be deleted without prompting!

  Example: cmemc project delete my_project

  Projects can be listed by using the 'cmemc project list' command.

Options:
  -a, --all   Delete all projects. This is a dangerous option, so use it with
              care.

  -h, --help  Show this message and exit.
```

4.3.3 Command: project export

```
Usage: cmemc project export [OPTIONS] [PROJECT_IDS]...

  Export project(s) to file(s).

  Projects can be exported with different export formats. The default type
  is a zip archive which includes meta data as well as dataset resources. If
```

```
  more than one project is exported, a file is created for each project. By
  default, these files are created in the current directory and with a
  descriptive name (see --template option default).

  Example: cmemc project export my_project

  Available projects can be listed by using the 'cmemc project list'
  command.

  You can use the template string to create subdirectories as well: cmemc
  config list | parallel -I% cmemc -c % project export --all -t
  "dump/{{connection}}/{{date}}-{{id}}.project"

Options:
  -a, --all                     Export all projects.
  -o, --overwrite               Overwrite existing files. This is a dangerous
                                option, so use it with care.

  --output-dir DIRECTORY        Directory, where the project files will be
                                created. This directory will be created as
                                well it does not exists.  [default: .]

  --type TEXT                   Type of the exported project file(s).
                                [default: xmlZip]

  -t, --filename-template TEXT  Template for the export file name(s). Possible
                                placeholders are (Jinja2): {{id}} (the project
                                ID), {{connection}} (from the --connection
                                option) and {{date}} (the current date as
                                YYYY-MM-DD). The file suffix will be appended.
                                Needed directories will be created.  [default:
                                {{id}}]

  -h, --help                    Show this message and exit.
```

4.3.4 Command: project import

```
Usage: cmemc project import [OPTIONS] FILE PROJECT_ID

  Import a project from a file.

  Example: cmemc project import my_project.zip my_project
```

```
Options:
  -h, --help  Show this message and exit.
```

4.3.5 Command: project list

```
Usage: cmemc project list [OPTIONS]

  List available projects.

  Outputs a list of project IDs which can be used as reference for the
  project create, delete, export and import commands.

Options:
  -h, --help  Show this message and exit.
```

4.4 Command group: query

```
Usage: cmemc query [OPTIONS] COMMAND [ARGS]...

  List, execute or open local and remote SPARQL queries.

  Queries are identified either by a file path, an URI from the query
  catalog or a shortened URI (qname, using a default namespace).

  In order to get a list of queries from the query catalog, use the list
  command. One or more queries can be executed one after the other with the
  execute command. With open command you can jump to the query editor in
  your browser.

  Queries can use a mustache like syntax to specify placeholder for
  parameter values (e.g. {{resourceUri}}). These parameter values need to be
  given as well, before the query can be executed (use the -p option).

Options:
  -h, --help  Show this message and exit.

Commands:
  execute  Execute queries which are loaded from files or the query catalog.
  list     List available queries from the catalog.
  open     Open queries in the editor of the query catalog in your browser.
```

4.4.1 Command: query execute

```
Usage: cmemc query execute [OPTIONS] QUERIES...

  Execute queries which are loaded from files or the query catalog.

  Queries are identified either by a file path, an URI from the query
  catalog, or a shortened URI (qname, using a default namespace).

  If multiple queries are executed one after the other, the first failing
  query stops the whole execution chain.

  Limitations: All optional parameters (e.g. accept, base64, ...) are
  provided for ALL queries in an execution chain. If you need different
  parameters for each query in a chain, run cmemc multiple times and use the
  logical operators && and || of your shell instead.

Options:
  --accept TEXT                   Accept header for the HTTP request(s).
                                  Setting this to 'default' means that cmemc
                                  uses an appropriate accept header for
                                  terminal output (text/csv for tables,
                                  text/turtle for graphs, * otherwise). Please
                                  refer to the Corporate Memory system manual
                                  for a list of accepted mime types.
                                  [default: default]

  --no-imports                    Graphs which include other graphs (using
                                  owl:imports) will bequeried as merged
                                  overall-graph. This flag disables
                                  thisdefault behaviour. The flag has no
                                  effect on update queries.

  --base64                        Enables base64 encoding of the query
                                  parameter for the SPARQL requests (the
                                  response is not touched). This can be useful
                                  in case there is an aggressive firewall
                                  between cmemc and Corporate Memory.

  -p, --parameter <TEXT TEXT>...  In case of a parameterized query
                                  (placeholders with the '{{key}}' syntax),
                                  this option fills all placeholder with a
                                  given value before the query is
                                  executed.Pairs of placeholder/value need to
```

```
                                  be given as a tuple 'KEY VALUE'. A key can
                                  be used only once.

  -h, --help                      Show this message and exit.
```

4.4.2 Command: query list

```
Usage: cmemc query list [OPTIONS]

  List available queries from the catalog.

  Outputs a list of query URIs which can be used as reference for the query
  execute command.

Options:
  --id-only   Lists only query identifier and no labels or other meta
              data.This is useful for piping the ids into other cmemc
              commands.

  -h, --help  Show this message and exit.
```

4.4.3 Command: query open

```
Usage: cmemc query open [OPTIONS] QUERIES...

  Open queries in the editor of the query catalog in your browser.

  With this command, you can open (remote) queries from the query catalog in
  the query editor in your browser (e.g. in order to change them). You can
  also load local query files into the query editor, in order to import them
  into the query catalog.

  The command accepts multiple query URIs or files which results in opening
  multiple browser tabs.

Options:
  -h, --help  Show this message and exit.
```

4.5 Command group: workflow

```
Usage: cmemc workflow [OPTIONS] COMMAND [ARGS]...

  List, execute, open or inspect workflows.

  Workflows are identified by a WORKFLOW_ID. The get a list of existing
  workflows, execute the list command or use tab-completion. The WORKFLOW_ID
  is a concatenation of an PROJECT_ID and a TASK_ID, such as "my-project:my-
  workflow".

Options:
  -h, --help  Show this message and exit.

Commands:
  execute  Execute workflow(s).
  list     List available workflow ids.
  open     Open a workflow in your browser.
  status   Get status information workflow(s).
```

4.5.1 Command: workflow execute

```
Usage: cmemc workflow execute [OPTIONS] [WORKFLOW_IDS]...

  Execute workflow(s).

  With this command, you can start one or more workflows at the same time or
  in a sequence, depending on the result of the predecessor.

  Executing a workflow can be done in two ways: Without --wait just sends
  the starting signal and does not look for the workflow and its result
  (fire and forget). Starting workflows in this way, starts all given
  workflows at the same time.

  The optional --wait option starts the workflows in the same way, but also
  polls the status of a workflow until it is finished. In case of an error
  of a workflow, the next workflow is not started.

Options:
  -a, --all                       Execute all available workflows.
  --wait                          Wait for one workflow to complete.
  --polling-interval INTEGER RANGE
                                  How many seconds to wait between status
                                  polls. Status polls are cheap, so a higher
```

```
                                  polling interval is most likely not needed.
                                  [default: 1]

  -h, --help                      Show this message and exit.
```

4.5.2 Command: workflow list

```
Usage: cmemc workflow list [OPTIONS]

  List available workflow ids.

Options:
  -h, --help  Show this message and exit.
```

4.5.3 Command: workflow open

```
Usage: cmemc workflow open [OPTIONS] WORKFLOW_ID

  Open a workflow in your browser.

Options:
  -h, --help  Show this message and exit.
```

4.5.4 Command: workflow status

```
Usage: cmemc workflow status [OPTIONS] [WORKFLOW_IDS]...

  Get status information workflow(s).

Options:
  --raw                      Output raw JSON info.
  --filter [running|failed]  Show only workflows of a specific status.
  -h, --help                 Show this message and exit.
```

4.6 Command group: workspace

```
Usage: cmemc workspace [OPTIONS] COMMAND [ARGS]...

  Import or export the workspace.
```

```
Options:
  -h, --help  Show this message and exit.

Commands:
  export  Export the complete workspace (all projects) to a ZIP file.
  import  Import the workspace from a file.
  reload  Reload the workspace from the backend.
```

4.6.1 Command: workspace export

```
Usage: cmemc workspace export [OPTIONS] [FILE]

  Export the complete workspace (all projects) to a ZIP file.

  Depending on the requested type, this ZIP contains either a turtle file
  for each project (type rdfTurtle) or a substructure of resource files and
  XML descriptions (type xmlZip).

  The file name is optional and will be generated with by the template if
  absent.

Options:
  -o, --overwrite               Overwrite existing files. This is a dangerous
                                option, so use it with care.

  --type TEXT                   Type of the exported workspace file.
                                [default: xmlZip]

  -t, --filename-template TEXT  Template for the export file name. Possible
                                placeholders are (Jinja2): {{connection}}
                                (from the --connection option) and {{date}}
                                (the current date as YYYY-MM-DD). The file
                                suffix will be appended. Needed directories
                                will be created.  [default:
                                {{date}}-{{connection}}.workspace]

  -h, --help                    Show this message and exit.
```

4.6.2 Command: workspace import

```
Usage: cmemc workspace import [OPTIONS] FILE
```

```
  Import the workspace from a file.

Options:
  --type TEXT  Type of the exported workspace file.  [default: xmlZip]
  -h, --help   Show this message and exit.
```

4.6.3 Command: workspace reload

```
Usage: cmemc workspace reload [OPTIONS]

  Reload the workspace from the backend.

Options:
  -h, --help  Show this message and exit.
```