eccenca
mastering complexity

Corporate Memory Control (cmemc)

v21.04

Manual

Contents

1 Introduction

This manual describes how to install and setup and use *eccenca Corporate Memory Control* (cmemc), the 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.

The main documentation resource for cmemc https://eccenca.com/go/cmemc.

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, edit and check configurations.
- List, create, delete, inspect datasets.
- List, import, export, delete or open graphs.
- List, import, export, create or delete projects.
- List, execute or open local and remote SPARQL queries.
- List, install, uninstall and open vocabularies.
- List, execute, open or inspect workflows.
- Import or 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.exe` - the PE32+ executable (console, x86-64), for Microsoft Windows, tested with Windows 10
- `cmemc_vXX.YY_Manual.pdf` - this document

2 Installation

cmemc can be installed using the python sources, using the release package or using the docker image.

2.1 Installation via pip

cmemc is available as an official pypi package[1] so installation can be done with pip:

```
$ pip install cmem-cmemc
```

2.2 Installation via release package

The cmemc distribution in the release package consists of stand alone binaries, so installation is not really 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.

2.2.1 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

---

[1] https://pypi.org/project/cmem-cmemc/

```
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[2].

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:

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

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

2.2.2 MacOS installation

There is no officially supported MacOS binary. Please use the installation with pip method.

As an alternative installation method, you can use brew:

```
brew install eccenca/main/cmemc
```

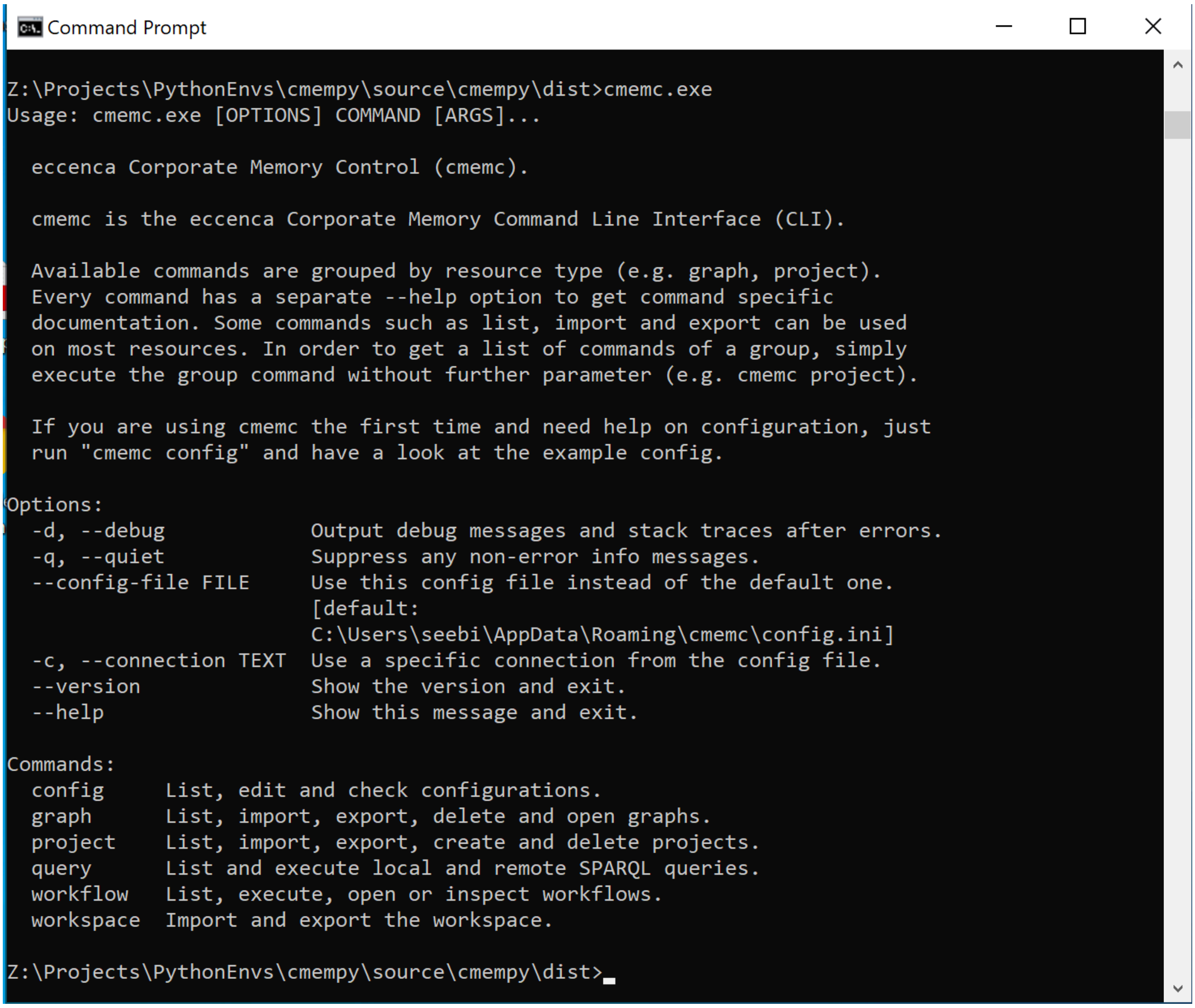
2.2.3 Windows installation

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

---

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



```
Z:\Projects\PythonEnvs\cmempy\source\cmempy\dist>cmemc.exe
Usage: cmemc.exe [OPTIONS] COMMAND [ARGS]...

  eccenca Corporate Memory Control (cmemc).

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

  Available commands are grouped by resource type (e.g. graph, project).
  Every command has a separate --help option to get command specific
  documentation. Some commands such as list, import and export can be used
  on most resources. In order to get a list of commands of a group, simply
  execute the group command without further parameter (e.g. cmemc project).

  If you are using cmemc the first time and need help on configuration, just
  run "cmemc config" and have a look at the example config.

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:
                         C:\Users\seebi\AppData\Roaming\cmemc\config.ini]
  -c, --connection TEXT  Use a specific connection from the config file.
  --version              Show the version and exit.
  --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.

Z:\Projects\PythonEnvs\cmempy\source\cmempy\dist>
```

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.
  dataset     List, create, delete, inspect datasets.
  graph       List, import, export, delete or open graphs.
  project     List, import, export, create or delete projects.
  query       List, execute or open local and remote SPARQL queries.
  vocabulary  List, install, uninstall and open vocabularies.
  workflow    List, execute, open or inspect workflows.
  workspace   Import or 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:

---

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

- `DI_API_ENDPOINT` - Data Integration API endpoint, default: `CMEM_BASE_URI/dataintegration`
- `DP_API_ENDPOINT` - Data Platform API endpoint, default: `CMEM_BASE_URI/dataplatform`
- `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`, only if `OAUTH_GRANT_TYPE` is `password`
- `OAUTH_PASSWORD` - Password to retrieve the token, default: `admin`, only if `OAUTH_GRANT_TYPE` is `password`
- `OAUTH_CLIENT_ID` - OAuth 2.0 client id, default: `cmem-service-account`
- `OAUTH_CLIENT_SECRET` - OAuth 2.0 client secret, default: `secret`, only if `OAUTH_GRANT_TYPE` is `client_credentials`
- `SSL_VERIFY` - Use SSL verification for requests to DP/DI default: `True`
- `OAUTH_AUTH_TOKEN` - a pre-fetched auth token, only if `OAUTH_GRANT_TYPE` is `prefetched_token`

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://ns.eccenca.com/example/data/vocabs/
https://vocab.eccenca.com/shacl/
urn:elds-backend-access-conditions-graph
https://ns.eccenca.com/data/queries/
```

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: admin

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

  Import bootstrap data, backup/restore workspace or get status.

  This command group consists of commands for setting up and configuring
  eccenca Corporate Memory.

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

Commands:
  bootstrap  Update/Import bootstrap data.
  showcase   Create showcase data.
  status     Output health and version information.
  token      Fetch and output an access token.
  workspace  Import, export and reload the project workspace.
```

4.1.1 Command: admin showcase

```
Usage: cmemc [OPTIONS]

  Create showcase data.

  This command creates a showcase scenario of multiple graphs including
  integration graphs, shapes, statement annotations etc.

  Note: There is currently no deletion mechanism for the showcase data, so
  you need to remove the showcase graphs manually (or just remove all
  graphs).

Options:
  --scale INTEGER  The scale factor provides a way to set the target size of
                   the scenario. A value of 10 results in around 40k triples,
                   a value of 50 in around 350k triples.  [default: 10]

  --create         Delete old showcase data if present and create new showcase
                   databased on the given scale factor.

  --delete         Delete existing showcase data if present.
  -h, --help       Show this message and exit.
```

4.1.2 Command: admin bootstrap

```
Usage: cmemc [OPTIONS]

  Update/Import bootstrap data.

  This command imports the bootstrap data needed for managing shapes, access
  conditions, the query catalog and the vocabulary catalog.

  Note: There is currently no deletion mechanism for the bootstrap data, so
  you need to remove the graphs manually (or just remove all graphs).

Options:
  --import    Delete existing bootstrap data if present and import bootstrap
              data which was delivered

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

4.1.3 Command: admin status

```
Usage: cmemc [OPTIONS]

  Output health and version information.

  This command outputs version and health information of the selected
  deployment. In addition to that, this command warns you if the target
  version of your cmemc client is newer than the version of your backend.

  To get status information of all configured deployments use this command
  in combination with xargs:

  cmemc config list | xargs -i cmemc -c {} admin status
```

```

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

4.1.4 Command: admin token

```
Usage: cmemc [OPTIONS]

  Fetch and output an access token.

  This command can be used to check for correct authentication as well as to
  use the token with wget / curl or similar standard tools:

  Example Usage: curl -H "Authorization: Bearer $(cmemc -c my admin token)"
  $(cmemc -c my config get DP_API_ENDPOINT)/api/custom/slug

  Please be aware that this command can reveal secrets, which you do not
  want to have in log files or on the screen.

Options:
  --raw       Outputs raw JSON. Note that this option will always try to fetch
              a new JSON token response. In case you are working with
              OAUTH_GRANT_TYPE=prefetched_token, this may lead to an error.

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

4.2 Command group: admin workspace

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

  Import, export and reload the project 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.
```

4.2.1 Command: admin workspace export

```
Usage: cmemc [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.
```

4.2.2 Command: admin workspace import

```
Usage: cmemc [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.
```

4.2.3 Command: admin workspace reload

```
Usage: cmemc [OPTIONS]

  Reload the workspace from the backend.

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

4.3 Command group: config

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

  List and edit configs as well as get config values.

  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

  Note that OAUTH_GRANT_TYPE can be either client_credentials, password or
  prefetched_token.

  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-standard DataPlatform location
  DI_API_ENDPOINT=URL - to point to a non-standard DataIntegration location
  OAUTH_TOKEN_URI=URL - to point to an external IdentityProvider location
  OAUTH_USER=username - only if OAUTH_GRANT_TYPE=password
  OAUTH_PASSWORD=password - only if OAUTH_GRANT_TYPE=password
  OAUTH_ACCESS_TOKEN=token - only if OAUTH_GRANT_TYPE=prefetched_token

  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:
  edit  Edit the user-scope configuration file.
  get   Get the value of a known cmemc configuration key.
  list  List configured connections.
```

4.3.1 Command: config list

```
Usage: cmemc [OPTIONS]

  List configured connections.

  This command lists all configured connections from the currently used
  config file.

  The connection identifier can be used with the --connection option in
  order to use a specific Corporate Memory instance.

  In order to apply commands on more than one instance, you need to use
  typical unix gear such as xargs or parallel:

  cmemc config list | xargs -I % sh -c 'cmemc -c % admin status'

  cmemc config list | parallel --jobs 5 cmemc -c {} admin status

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

4.3.2 Command: config edit

```
Usage: cmemc [OPTIONS]

  Edit the user-scope configuration file.

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

4.3.3 Command: config get

```
Usage: cmemc [OPTIONS] [CMEM_BASE_URI|SSL_VERIFY|REQUESTS_CA_BUNDLE|DP_API_END
             POINT|DI_API_ENDPOINT|OAUTH_TOKEN_URI|OAUTH_GRANT_TYPE|OAUTH_USER
             |OAUTH_PASSWORD|OAUTH_CLIENT_ID|OAUTH_CLIENT_SECRET|OAUTH_ACCESS_
             TOKEN]

  Get the value of a known cmemc configuration key.

  In order to automate processes such as fetching custom API data from
  multiple Corporate Memory instances, this command provides a way to get
  the value of a cmemc configuration key for the selected deployment.

  Example Usage: curl -H "Authorization: Bearer $(cmemc -c my admin token)"
  $(cmemc -c my config get DP_API_ENDPOINT)/api/custom/slug

  The commands returns with exit code 1 if the config key is not used in the
  current configuration.

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

4.4 Command group: dataset

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

  List, create, delete, inspect, up-/download or open datasets.

  This command group allows for managing workspace datasets as well as
  dataset file resources. Datasets can be created and deleted. File
  resources can be uploaded and downloaded. Details of dataset parameter can
  be listed with inspect.

  Datasets are identified with a combined key of the project ID and the
  project internal dataset ID (e.g: my-project:my-dataset). To get a list of
  datasets, use the list command.

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

Commands:
  create    Create a dataset.
  delete    Delete datasets.
  download  Download the resource file of a dataset.
  inspect   Display meta data of a dataset.
  list      List available datasets.
  open      Open datasets in the browser.
  upload    Upload a resource file to a dataset.
```

4.4.1 Command: dataset list

```
Usage: cmemc [OPTIONS]

  List available datasets.

  Outputs a list of datasets IDs which can be used as reference for the
  dataset create and delete commands.

Options:
  --project TEXT  The project, from which you want to list the datasets.
                  Project IDs can be listed with the 'project list' command.

  --raw           Outputs raw JSON objects of dataset search API response.
  --id-only       Lists only dataset 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.
```

4.4.2 Command: dataset delete

```
Usage: cmemc [OPTIONS] [DATASET_IDS]...

  Delete datasets.

  This deletes existing datasets in integration projects from Corporate
  Memory. Datasets will be deleted without prompting! Dataset resources will
  not be deleted.

  Example: cmemc dataset delete my_project:my_dataset

  Datasets can be listed by using the 'cmemc dataset list' command.

Options:
  -a, --all       Delete all datasets. This is a dangerous option, so use it
```

```
                  with care.

  --project TEXT  In combination with the '--all' flag, this option allows for
                  deletion of all datasets of a certain project. The behaviour
                  is similar to the 'dataset list --project' command.

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

4.4.3 Command: dataset download

```
Usage: cmemc [OPTIONS] DATASET_ID OUTPUT_PATH

  Download the resource file of a dataset.

  This command downloads the file resource of a dataset to your local file
  system or to standard out (-). Note that this is not possible for dataset
  types such as Knowledge Graph (eccencaDataplatform) or SQL endpoint
  (sqlEndpoint).

  Without providing an output path, the output file name will be the same as
  the remote file resource.

  Datasets can be listed by using the 'cmemc dataset list' command.

Options:
  --replace   Replace existing files. This is a dangerous option, so use it
              with care.

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

4.4.4 Command: dataset upload

```
Usage: cmemc [OPTIONS] DATASET_ID INPUT_PATH

  Upload a resource file to a dataset.

  This command uploads a file to a dataset. The content of the uploaded file
  replaces the remote file resource. The name of the remote file resource is
  not changed.

  Warning: If the remote file resource is used in more than one dataset, the
  other datasets are also affected by this command.
  Warning: The content of the uploaded file is not tested, so uploading a
  json file to an xml dataset will result in errors.

  Datasets can be listed by using the 'cmemc dataset list' command.

  Example: cmemc dataset upload cmem:my-dataset new-file.csv

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

4.4.5 Command: dataset inspect

```
Usage: cmemc [OPTIONS] DATASET_ID

  Display meta data of a dataset.

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

4.4.6 Command: dataset create

```
Usage: cmemc [OPTIONS] [DATASET_FILE]

  Create a dataset.

  Datasets are created in projects and can have associated file resources.
  Each dataset has a type (such as 'csv') and a list of parameter which can
  change or specify the dataset behaviour.

  To get more information on possible dataset types and parameter on these
  types, use the '--help-types' and '--help-parameter' options.

  Example: cmemc dataset create --project my-project --type csv my-file.csv

Options:
  -t, --type TEXT                 The dataset type of the dataset to create.
                                  Example types are 'csv','json' and
                                  'eccencaDataPlatform' (-> Knowledge Graph).

  --project TEXT                  The project, where you want to create the
```

```
                                  dataset in. If there is only one project in
                                  the workspace, this option can be omitted.

  -p, --parameter <TEXT TEXT>...  A set of key/value pairs. Each dataset type
                                  has different parameters (such as charset,
                                  arraySeparator, ignoreBadLines, ...). In
                                  order to get a list of possible parameter,
                                  use the'--help-parameter' option.

  --replace                       Replace remote file resources in case there
                                  already exists a file with the same name.

  --id TEXT                       The dataset ID of the dataset to create. The
                                  dataset ID will be automatically created in
                                  case it is not present.

  --help-types                    Lists all possible dataset types on given
                                  Corporate Memory instance. Note that this
                                  option already needs access to the instance.

  --help-parameter                Lists all possible (optional and mandatory)
                                  parameter for a dataset type. Note that this
                                  option already needs access to the instance.

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

4.4.7 Command: dataset open

```
Usage: cmemc [OPTIONS] DATASET_IDS...

  Open datasets in the browser.

  With this command, you can open a dataset in the workspace in your
  browser.

  The command accepts multiple dataset IDs which results in opening multiple
  browser tabs.

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

4.5 Command group: graph

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

  List, import, export, delete, count, tree 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.
  tree    Show graph tree(s) of the owl:imports hierarchy.
```

4.5.1 Command: graph count

```
Usage: cmemc [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.
```

4.5.2 Command: graph tree

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

  Show graph tree(s) of the owl:imports hierarchy.

  You can can output one or more trees of the import hierarchy.

  Imported graphs which do not exists are shown as [missing: IRI]. Imported
  graphs which will result in an import cycle are shown as [ignored: IRI].
  Otherwise each graph is shown with label and IRI or only as IRI (in case
  of --id-only usage).

Options:
  -a, --all   Show tree of all (readable) graphs.
  --raw       Outputs raw JSON of the graph importTree API response.
  --id-only   Shows only graph IRIs in the tree and no labels.
  -h, --help  Show this message and exit.
```

4.5.3 Command: graph list

```
Usage: cmemc [OPTIONS]

  List accessible graphs.

Options:
  --raw                        Outputs raw JSON of the graphs list API response.
  --id-only                    Lists only graph identifier (IRIs) and no labels
                               or other meta data. This is useful for piping the
                               IRIs into other commands.

  --filter <CHOICE TEXT>...    Filter graphs based on effective access
                               conditions or import closure. First parameter
                               CHOICE can be 'access' or 'imported-by'. The
                               second parameter can be 'readonly' or 'writeable'
                               in case of 'access' or any readable graph in case
                               of 'imported-by'.

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

4.5.4 Command: graph export

```
Usage: cmemc [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 .ttl files will be
  created for each graph.
```

```
Options:
  -a, --all                       Export all readable graphs.
  --include-imports               Export selected graph(s) and all graphs
                                  which are imported from these selected
                                  graph(s).

  --create-catalog                In addition to the .ttl and .graph files,
                                  cmemc will create an XML catalog file
                                  (catalog-v001.xml) which can be used by
                                  applications such as Protégé.

  --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}} - sha256
                                  hash of the graph IRI, {{iriname}} - graph
                                  IRI converted to filename, {{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}}]

  --mime-type [application/n-triples|text/turtle]
                                  Define the requested mime type  [default:
                                  application/n-triples]

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

4.5.5 Command: graph delete

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

  Delete graph(s) from the store.

Options:
  -a, --all          Delete all writeable graphs.
  --include-imports  Delete selected graph(s) and all writeable graphs which
                     are imported from these selected graph(s).

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

4.5.6 Command: graph import

```
Usage: cmemc [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.
```

4.5.7 Command: graph open

```
Usage: cmemc [OPTIONS] IRI

  Open / explore a graph in the browser.

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

4.6 Command group: project

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

  List, import, export, create, delete or open 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 or directory.
  list    List available projects.
  open    Open projects in the browser.
```

4.6.1 Command: project open

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

  Open projects in the browser.

  With this command, you can open a project in the workspace in your browser
  to change them.

  The command accepts multiple projects IDs which results in opening
  multiple browser tabs.

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

4.6.2 Command: project list

```
Usage: cmemc [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:
  --raw       Outputs raw JSON.
  --id-only   Lists only project identifier and no labels or other meta data.
              This is useful for piping the IDs into other commands.

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

4.6.3 Command: project export

```
Usage: cmemc [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          The base directory, where the project files
                                  will be created. If this directory does not
                                  exist, it will be silently created.  [default:
                                  .]

  --type TEXT                     Type of the exported project file(s). Use the
                                  --help-types option or tab completion to see a
                                  list of possible types.  [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:
                                  {{date}}-{{connection}}-{{id}}.project]

  --extract                       Export projects to a directory structure
                                  instead of a ZIP archive. Note that the
                                  --filename-template option is ignored here.
                                  Instead, a sub-directory per exported project
                                  is created under the output directory. Also
                                  note that not all export types are
                                  extractable.
    --help-types                  Lists all possible export types.
    -h, --help                    Show this message and exit.
```

4.6.4 Command: project import

```
Usage: cmemc [OPTIONS] PATH PROJECT_ID

  Import a project from a file or directory.

  Example: cmemc project import my_project.zip my_project

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

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

4.6.5 Command: project delete

```
Usage: cmemc [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.
```

4.6.6 Command: project create

```
Usage: cmemc [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.
```

4.7 Command group: query

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

  List, execute, get status or open 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.
  status   Get status information of executed and running queries.
```

4.7.1 Command: query execute

```
Usage: cmemc [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 be queried as merged
                                  overall-graph. This flag disables this
                                  default 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.

  --limit INTEGER                 Override or set the LIMIT in the executed
```

```
                                   SELECT query. Note that this option will
                                   never give you more results than the LIMIT
                                   given in the query itself.

  --offset INTEGER                 Override or set the OFFSET in the executed
                                   SELECT query.

  --distinct                       Override the SELECT query by make the result
                                   set DISTINCT.

  --timeout INTEGER                Set max execution time for query evaluation
                                   (in milliseconds).

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

4.7.2 Command: query list

```
Usage: cmemc [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.
```

4.7.3 Command: query open

```
Usage: cmemc [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.
```

4.7.4 Command: query status

```
Usage: cmemc [OPTIONS] [QUERY_UUID]

  Get status information of executed and running queries.

  With this command, you can access the latest executed SPARQL queries on
  the DataPlatform. These queries are identified by UUIDs and listed ordered
  by starting timestamp.

  You can filter queries based on status and runtime in order to investigate
  slow queries. In addition to that, you can get the details of a specific
  query by using the UUID as a parameter.

  Known Issue: Timeouted queries are currently shown as running queries
  (means: they have no runtime and no ending timestamp). This will be fixed
  in a future release.

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.

  --raw                      Outputs raw JSON response of the query status
                             API.

  --filter <CHOICE TEXT>...  Filter queries based on execution status and
                             time. First parameter CHOICE can be 'status' or
                             'slower-than'. The second parameter has to be a
                             status  in case of 'status' or a time in
                             miliseconds in case of 'slower-than' filter.

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

4.8 Command group: vocabulary

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

```

  List, (un-)install, import or open vocabs / manage cache.

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

Commands:
  cache      List und update the vocabulary cache.
  import     Import a turtle file as a vocabulary.
  install    Install one or more vocabularies from the catalog.
  list       Output a list of vocabularies.
  open       Open / explore a vocabulary graph in the browser.
  uninstall  Uninstall one or more vocabularies.
```

4.8.1 Command: vocabulary open

```
Usage: cmemc [OPTIONS] IRI

  Open / explore a vocabulary graph in the browser.

  Vocabularies are identified by their graph IRI. Installed vocabularies can
  be listed with the "vocabulary list" command.

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

4.8.2 Command: vocabulary list

```
Usage: cmemc [OPTIONS]

  Output a list of vocabularies.

  Vocabularies are graphs (see 'cmemc graph' command group) which consists
  of class and property descriptions.

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

  --filter [all|installed|installable]
```

```
                                  Filter list based on status.  [default:
                                  installed]

  --raw                           Outputs raw JSON.
  -h, --help                      Show this message and exit.
```

4.8.3 Command: vocabulary install

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

  Install one or more vocabularies from the catalog.

  Vocabularies are identified by their graph IRI. Installable vocabularies
  can be listed with the "vocabulary list --filter installable" command.

Options:
  -a, --all   Install all vocabularies from the catalog.
  -h, --help  Show this message and exit.
```

4.8.4 Command: vocabulary uninstall

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

  Uninstall one or more vocabularies.

  Vocabularies are identified by their graph IRI. Already installed
  vocabularies can be listed with the "vocabulary list --filter installed"
  command.

Options:
  -a, --all   Uninstall all installed vocabularies.
  -h, --help  Show this message and exit.
```

4.8.5 Command: vocabulary import

```
Usage: cmemc [OPTIONS] FILE

  Import a turtle file as a vocabulary.

  With this command, you can import a local ontology file as a named graph.
  and create a corresponding vocabulary catalog entry.

  The uploaded ontology file is analysed locally in order to discover the
  named graph and the prefix declaration. This requires an OWL ontology
  description which correctly uses the vann:preferredNamespacePrefix and
  vann:preferredNamespaceUri properties.

Options:
  --replace   Replace (overwrite) existing vocabulary, if present.
  -h, --help  Show this message and exit.
```

4.9 Command group: vocabulary cache

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

  List und update the vocabulary cache.

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

Commands:
  list    Output the content of the global vocabulary cache.
  update  Reload / updates the data integration cache for a vocabulary.
```

4.9.1 Command: vocabulary cache update

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

  Reload / updates the data integration cache for a vocabulary.

Options:
  -a, --all   Update cache for all installed vocabularies.
  -h, --help  Show this message and exit.
```

4.9.2 Command: vocabulary cache list

```
Usage: cmemc [OPTIONS]

  Output the content of the global vocabulary cache.

Options:
  --id-only   Lists only vocabulary term identifier (IRIs) and no labels or
```

```
               other meta data. This is useful for piping the ids into other
               cmemc commands.

  --raw        Outputs raw JSON.
  -h, --help   Show this message and exit.
```

4.10 Command group: workflow

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

  List, execute, status or open (io) 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).
  io       Execute a workflow with file input/output.
  list     List available workflow ids.
  open     Open a workflow in your browser.
  status   Get status information of workflow(s).
```

4.10.1 Command: workflow execute

```
Usage: cmemc [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 until all executed workflows are
                                  completed.

  --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.
```

4.10.2 Command: workflow io

```
Usage: cmemc [OPTIONS] WORKFLOW_ID

  Execute a workflow with file input/output.

  With this command, you can execute a workflow that uses variable datasets
  as input, output or for configuration. Use the input parameter to feed
  data into the workflow. Likewise use output for retrieval of the workflow
  result. Workflows without a variable dataset will throw an error.

Options:
  -i, --input FILE                From which file the input is taken: note
                                  that the maximum file size to upload is
                                  limited to a server configured value. If the
                                  workflow has no defined variable input
                                  dataset, this can be ignored.

  -o, --output FILE               To which file the result is written to: use
                                  '-' in order to output the result to stdout.
                                  If the workflow has no defined variable
                                  output dataset, this can be ignored. Please
                                  note that the io command will not warn you
                                  on overwriting existing output files.

  --input-mimetype [guess|application/xml|application/json|text/csv]
                                  Which input format should be processed: If
                                  not given, cmemc will try to guess the mime
                                  type based on the file extension or will
                                  fail

  --output-mimetype
      [guess|application/xml|application/json|application/n-triples|application/vnd.openxmlformats-officedo
                                  Which output format should be requested: If
                                  not given, cmemc will try to guess the mime
                                  type based on the file extension or will
                                  fail. In case of an output to stdout, a
                                  default mime type will be used (currently
                                  xml).

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

4.10.3 Command: workflow list

```
Usage: cmemc [OPTIONS]

  List available workflow ids.

Options:
  --raw                      Outputs raw JSON objects of workflow task search
                             API response.

  --id-only                  Lists only workflow identifier and no labels or
                             other meta data. This is useful for piping the
                             IDs into other commands.

  --filter <CHOICE TEXT>...  Filter workflows based on project or suitability
                             for the io command .First parameter CHOICE can be
                             'project' or 'io'. The second parameter has to be
                             a project ID in case of 'project' or 'input-
                             only|output-only|input-output|any' in case of
                             'io' filter.

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

4.10.4 Command: workflow status

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

```
  Get status information of workflow(s).

Options:
  --project TEXT                  The project, from which you want to list the
                                  workflows. Project IDs can be listed with
                                  the 'project list' command.

  --raw                           Output raw JSON info.
  --filter [Idle|Not executed|Finished|Cancelled|Failed|Successful|Canceling|Running|Waiting]
                                  Show only workflows of a specific status.
  -h, --help                      Show this message and exit.
```

4.10.5 Command: workflow open

```
Usage: cmemc [OPTIONS] WORKFLOW_ID

  Open a workflow in your browser.

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