



Corporate Memory Control (cmemc)

v22.2

Manual



Contents

1	Introduction	5
1.1	About eccenca Corporate Memory Control (cmemc)	5
1.2	Scope of delivery	6
2	Installation	7
2.1	Installation via pip	7
2.2	Installation via release package	7
2.2.1	Linux / MacOS installation	7
2.2.2	Windows installation	8
3	Configuration	9
4	Command Reference	12
4.1	Command group: admin	12
4.1.1	Command: admin status	12
4.1.2	Command: admin token	13
4.2	Command group: admin metrics	14
4.2.1	Command: admin metrics get	14
4.2.2	Command: admin metrics inspect	15
4.2.3	Command: admin metrics list	15
4.3	Command group: admin workspace	16
4.3.1	Command: admin workspace export	16
4.3.2	Command: admin workspace import	17
4.3.3	Command: admin workspace reload	17
4.4	Command group: admin workspace python	17
4.4.1	Command: admin workspace python install	18
4.4.2	Command: admin workspace python uninstall	18
4.4.3	Command: admin workspace python list	19
4.4.4	Command: admin workspace python list-plugins	19
4.5	Command group: admin store	19
4.5.1	Command: admin store showcase	20
4.5.2	Command: admin store bootstrap	20
4.5.3	Command: admin store export	21
4.5.4	Command: admin store import	21
4.6	Command group: config	22
4.6.1	Command: config list	23
4.6.2	Command: config edit	24

4.6.3	Command: config get	24
4.6.4	Command: config eval	24
4.7	Command group: dataset	25
4.7.1	Command: dataset list	26
4.7.2	Command: dataset delete	26
4.7.3	Command: dataset download	27
4.7.4	Command: dataset upload	27
4.7.5	Command: dataset inspect	28
4.7.6	Command: dataset create	28
4.7.7	Command: dataset open	29
4.8	Command group: dataset resource	30
4.8.1	Command: dataset resource list	30
4.8.2	Command: dataset resource delete	31
4.8.3	Command: dataset resource inspect	31
4.8.4	Command: dataset resource usage	31
4.9	Command group: graph	32
4.9.1	Command: graph count	32
4.9.2	Command: graph tree	32
4.9.3	Command: graph list	33
4.9.4	Command: graph export	33
4.9.5	Command: graph delete	34
4.9.6	Command: graph import	35
4.9.7	Command: graph open	35
4.10	Command group: project	35
4.10.1	Command: project open	36
4.10.2	Command: project list	36
4.10.3	Command: project export	37
4.10.4	Command: project import	38
4.10.5	Command: project delete	38
4.10.6	Command: project create	39
4.10.7	Command: project reload	39
4.11	Command group: query	40
4.11.1	Command: query execute	40
4.11.2	Command: query list	42
4.11.3	Command: query open	42
4.11.4	Command: query status	43
4.11.5	Command: query replay	43
4.11.6	Command: query cancel	44
4.12	Command group: vocabulary	44
4.12.1	Command: vocabulary open	45
4.12.2	Command: vocabulary list	45
4.12.3	Command: vocabulary install	46
4.12.4	Command: vocabulary uninstall	46
4.12.5	Command: vocabulary import	46

4.13 Command group: vocabulary cache	47
4.13.1 Command: vocabulary cache update	47
4.13.2 Command: vocabulary cache list	48
4.14 Command group: workflow	48
4.14.1 Command: workflow execute	48
4.14.2 Command: workflow io	49
4.14.3 Command: workflow list	50
4.14.4 Command: workflow status	51
4.14.5 Command: workflow open	51
4.15 Command group: workflow scheduler	51
4.15.1 Command: workflow scheduler open	52
4.15.2 Command: workflow scheduler list	52
4.15.3 Command: workflow scheduler inspect	53
4.15.4 Command: workflow scheduler disable	53
4.15.5 Command: workflow scheduler enable	53

1 Introduction

This manual describes, how to install, 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 and 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 is <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 an open source python package.

Main features of cmemc include:

- List, edit and check configurations.
- List, create, delete, inspect datasets as well as dataset resources.
- List, import, export, delete or open graphs.
- List, import, export, create or delete Build projects.
- List, execute, replay or open local and remote SPARQL queries.
- List, install, uninstall, import and open vocabularies.
- List, execute, open or inspect workflows and workflow schedulers.
- Import or export whole Build workspaces and graph stores.
- List, get or inspect server metrics.

1.2 Scope of delivery

The cmemc release package consists of the following files:

- `cmem_cmemc-vXX.YY.tar.gz` - the source package of cmemc
- `cmem_cmempy-vXX.YY.tar.gz` - the source package of cmempy (the used python API to access Corporate Memory)
- `cmemc_vXX.YY_Manual.pdf` - the cmemc documentation manual (this document)
- `cmemc_vXX.YY_Manual.ttl` - the cmemc documentation as structured data (RDF graph)
- `requirements.txt` - additional requirements needed by cmemc

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 package¹, so installation can be done with pip or pipx²:

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

```
$ pipx install cmem-cmemc
```

2.2 Installation via release package

2.2.1 Linux / MacOS installation

The cmemc distribution in the release package consists of source package which can be installed with pip as well.

The following script demonstrates how to install cmemc from these files:

```
$ pip install -r requirements.txt
...
$ pip install cmem_cmempy-v22.1.tar.gz
...
$ pip install cmem_cmemc-v22.1.tar.gz
```

Finally, test your installation.

```
$ cmemc --version
cmemc, version 22.1
```

¹<https://pypi.org/project/cmem-cmemc/>

²<https://pypa.github.io/pipx/>

In case you are using bash or zsh as your shell, you should enable tab completion for cmemc. This is documented on the [click framework homepage](https://click.palletsprojects.com/en/7.x/bashcomplete/#activation)³.

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` in order to enable completion per default.

2.2.2 Windows installation

The installation for Windows is similar once you have installed python from the store.

³<https://click.palletsprojects.com/en/7.x/bashcomplete/#activation>

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`.

```
$ 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 and group has a separate --help screen
for detailed documentation. In order to see possible commands in a group,
simply execute the group command without further parameter (e.g. cmemc
project).

If your terminal supports colors, these coloring rules are applied: Groups
are colored in white; Commands which change data are colored in red; all
other commands as well as options are colored in green.

Please also have a look at the cmemc online documentation:

https://eccenca.com/go/cmemc
```

```
cmemc is © 2022 eccenca GmbH, licensed under the Apache License 2.0.
```

Options:

```
-c, --connection TEXT  Use a specific connection from the config file.
--config-file FILE     Use this config file instead of the default one.
                        [default: /Users/seebi/Library/Application
                        Support/cmenc/config.ini]

-q, --quiet            Suppress any non-error info messages.
-d, --debug            Output debug messages and stack traces after errors.
--version              Show the version and exit.
-h, --help             Show this message and exit.
```

Commands:

```
admin      Import bootstrap data, backup/restore workspace or get status.
config     List and edit configs as well as get config values.
dataset    List, create, delete, inspect, up-/download or open datasets.
graph      List, import, export, delete, count, tree or open graphs.
project    List, import, export, create, delete or open projects.
query      List, execute, get status or open SPARQL queries.
vocabulary List, (un-)install, import or open vocabs / manage cache.
workflow   List, execute, status or open (io) workflows.
```

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

```
$ cmemc config edit
Open editor for config file /home/user/.config/cmenc/config.ini
```

The rules for the config file are similar to a Windows INI file and are explained in detail at [docs.python.org](https://docs.python.org/3/library/configparser.html)¹. Here is a 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-`

¹<https://docs.python.org/3/library/configparser.html>

`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 same 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`
- `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:

```
$ cmemc -c my-local graph list
```

Graph IRI	Type	Label
urn:elds-backend-access-conditions-graph	void:Dataset	CMEM Access Conditions
https://ns.eccenca.com/data/config/	void:Dataset	CMEM Configuration
https://ns.eccenca.com/data/queries/	void:Dataset	CMEM Query Catalog
https://vocab.eccenca.com/shacl/	void:Dataset	CMEM Shapes Catalog
https://ns.eccenca.com/example/data/vocabs/	void:Dataset	CMEM Vocabulary Catalog

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

4 Command 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:

metrics	List and get metrics.
status	Output health and version information.
store	Import, export and bootstrap the knowledge graph store.
token	Fetch and output an access token.
workspace	Import, export and reload the project workspace.

4.1.1 Command: admin status

Usage: cmemc [OPTIONS]

Output health and version information.

This command outputs version and health information of the selected deployment. If the version information can not be retrieved, UNKNOWN shown.

In addition to that, this command warns you if the target version of your cmemc client is newer than the version of your backend and if the ShapeCatalog has a different version than your DataPlatform component.

To get status information of all configured deployments use this command in combination with parallel.

Example: `cmemc config list | parallel --ctag cmemc -c {} admin status`

Options:

- `--key TEXT` Get only specific key(s) from the status / info output. There are two special keys available: 'all' will list all available keys in the table, 'overall.healthy' with result in UP in case all health flags are UP as well (otherwise DOWN).
- `--enforce-table` A single value with `--key` will be returned as plain text instead of a table with one row and the header. This default behaviour allows for more easy integration with scripts. This flag enforces the use of tabular output, even for single row tables.
- `--raw` Outputs combined raw JSON output of the health/info endpoints.
- `-h, --help` Show this message and exit.

4.1.2 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: `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 `OAuthGrantType=prefetched_token`, this may lead to an error.

```
--decode    Decode the access token and outputs the raw JSON. Note that the
             access token is only decoded and esp. not validated.
```

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

4.2 Command group: admin metrics

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

List and get metrics.

This command group consists of commands for reading and listing internal monitoring metrics of eccenca Corporate Memory. A deployment consists of multiple jobs (e.g. DP, DI), which provide multiple metric families on an endpoint.

Each metric family can consist of different samples identified by labels with a name and a value (dimensions). A metric has a specific type (counter, gauge, summary and histogram) and additional metadata.

Please have a look at https://prometheus.io/docs/concepts/data_model/ for further details.

Options:

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

Commands:

```
get        Get sample data of a metric.
inspect    Inspect a metric.
list       List metrics for a specific job.
```

4.2.1 Command: admin metrics get

Usage: cmemc [OPTIONS] METRIC_ID

Get sample data of a metric.

A metric of a specific job is identified by a metric ID. Possible metric IDs of a job can be retrieved with the `metrics list` command. A metric can contain multiple samples. These samples are distinguished by labels (name and value).

Options:

<code>--job [DP]</code>	The job from which the metrics data is fetched. [default: DP]
<code>--filter <TEXT TEXT>...</code>	A set of label name/value pairs in order to filter the samples of the requested metric family. Each metric has a different set of labels with different values. In order to get a list of possible label names and values, use the command without this option. The label names are then shown as column headers and label values as cell values of this column.
<code>--enforce-table</code>	A single sample value will be returned as plain text instead of the normal table. This allows for more easy integration with scripts. This flag enforces the use of tabular output, even for single row tables.
<code>--raw</code>	Outputs raw prometheus sample classes.
<code>-h, --help</code>	Show this message and exit.

4.2.2 Command: admin metrics inspect

Usage: `cmemc [OPTIONS] METRIC_ID`

Inspect a metric.

This command outputs the data of a metric. The first table includes basic meta data about the metric. The second table includes sample labels and values.

Options:

<code>--job [DP]</code>	The job from which the metrics data is fetched.	[default: DP]
<code>--raw</code>	Outputs raw JSON of the table data.	
<code>-h, --help</code>	Show this message and exit.	

4.2.3 Command: admin metrics list

Usage: `cmemc [OPTIONS]`

List metrics for a specific job.

For each metric, the output table shows the metric ID, the type of the metric, a count of how many labels (label names) are describing the samples (L) and a count of how many samples are currently available for a metric (S).

Options:

--job [DP] The job from which the metrics data is fetched. [default: DP]
--id-only Lists metric identifier only. This is useful for piping the IDs into other commands.

--raw Outputs (sorted) JSON dict, parsed from the metrics API output.
-h, --help Show this message and exit.

4.3 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.
python List, install, or uninstall python packages.
reload Reload the workspace from the backend.

4.3.1 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:

<code>-o, --overwrite</code>	Overwrite existing files. This is a dangerous option, so use it with care.
<code>--type TEXT</code>	Type of the exported workspace file. [default: xmlZip]
<code>-t, --filename-template TEXT</code>	Template for the export file name. Possible placeholders are (Jinja2): <code>{{connection}}</code> (from the <code>--connection</code> option) and <code>{{date}}</code> (the current date as YYYY-MM-DD). The file suffix will be appended. Needed directories will be created. [default: <code>{{date}}-{{connection}}.workspace</code>]
<code>-h, --help</code>	Show this message and exit.

4.3.2 Command: admin workspace import

Usage: `cmemc [OPTIONS] FILE`

Import the workspace from a file.

Options:

<code>--type TEXT</code>	Type of the exported workspace file. [default: xmlZip]
<code>-h, --help</code>	Show this message and exit.

4.3.3 Command: admin workspace reload

Usage: `cmemc [OPTIONS]`

Reload the workspace from the backend.

Options:

<code>-h, --help</code>	Show this message and exit.
-------------------------	-----------------------------

4.4 Command group: admin workspace python

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

List, install, or uninstall python packages.

Python packages are used to extend the DataIntegration workspace with python plugins. To get a list of installed packages, execute the list command.

Warning: Installing packages from unknown sources is not recommended. Plugins are not verified for malicious code.

Options:

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

Commands:

install	Install a python package to the workspace.
list	List installed python packages.
list-plugins	List installed workspace plugins.
uninstall	Uninstall a python package from the workspace.

4.4.1 Command: admin workspace python install

Usage: cmemc [OPTIONS] PACKAGE

Install a python package to the workspace.

This command is basically a `pip install` in the remote python environment.

You can install a package by uploading a source distribution .tar.gz file, or by uploading a build distribution .whl file, or by specifying a package name, more precisely, a pip requirement specifier with a package name available on pypi.org (e.g. `requests==2.27.1`).

Options:

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

4.4.2 Command: admin workspace python uninstall

Usage: cmemc [OPTIONS] PACKAGE_NAME

Uninstall a python package from the workspace.

This command is basically a `pip uninstall` in the remote python environment.

Options:

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

4.4.3 Command: admin workspace python list

Usage: cmemc [OPTIONS]

List installed python packages.

This command is basically a `pip list` in the remote python environment.

It outputs a table of python package identifiers with version information.

Options:

--raw Outputs raw JSON.

--id-only Lists only package identifier. This is useful for piping the IDs into other commands.

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

4.4.4 Command: admin workspace python list-plugins

Usage: cmemc [OPTIONS]

List installed workspace plugins.

This commands lists all discovered plugins.

Note: The plugin discovery is restricted to package prefix (`cmem-`).

Options:

--raw Outputs raw JSON.

--id-only Lists only plugin identifier.

--package-id-only Lists only plugin package identifier.

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

4.5 Command group: admin store

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

Import, export and bootstrap the knowledge graph store.

This command group consist of commands to administrate the knowledge graph store as a whole.

Options:

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

Commands:

bootstrap Update/Import bootstrap data.

export Backup all knowledge graphs to a ZIP archive.

import Restore graphs from a ZIP archive.

showcase Create showcase data.

4.5.1 Command: admin store 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.

4.5.2 Command: admin store 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.

4.5.3 Command: admin store export

Usage: cmemc [OPTIONS] BACKUP_FILE

Backup all knowledge graphs to a ZIP archive.

The backup file is a ZIP archive containing all knowledge graphs as Turtle files + configuration file for each graph.

This command will create lots of load on the server. It can take a long time to complete.

Options:

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

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

4.5.4 Command: admin store import

Usage: cmemc [OPTIONS] BACKUP_FILE

Restore graphs from a ZIP archive.

The backup file is a ZIP archive containing all knowledge graphs as Turtle files + configuration file for each graph.

The command will load a single backup ZIP archive into the triple store,

by replacing all graphs with the content of the Turtle files in the archive and deleting all graphs which are not in the archive.

This command will create lots of load on the server. It can take a long time to complete. The backup file will be transferred to the server, then unzipped and imported graph by graph. After the initial transfer, the network connection is not used anymore, so it will be closed by proxies sometimes. This does not mean that the import failed.

Options:

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

4.6 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 get credential information from an external process, you can use the parameter `OAuth_PASSWORD_PROCESS`, `OAuth_CLIENT_SECRET_PROCESS` and `OAuth_ACCESS_TOKEN_PROCESS` to setup an external executable.

```
OAuth_CLIENT_SECRET_PROCESS=/path/to/getpass.sh
OAuth_PASSWORD_PROCESS=["getpass.sh", "parameter1", "parameter2"]
```

The credential executable can use the cmemc environment for fetching the credential (e.g. `CMEM_BASE_URI` and `OAuth_USER`). If the credential executable is not given with a full path, cmemc will look into your environment `PATH` for something which can be executed. The configured process needs to return the credential on the first line of stdout. In addition to that, the process needs to exit with exit code 0 (without failure). There are examples available in the online manual.

Options:

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

Commands:

```
edit  Edit the user-scope configuration file.
eval  Export all configuration values of a configuration for evaluation.
get   Get the value of a known cmemc configuration key.
list  List configured connections.
```

4.6.1 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`.

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

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

Options:

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

4.6.2 Command: config edit

Usage: cmemc [OPTIONS]

Edit the user-scope configuration file.

Options:

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

4.6.3 Command: config get

Usage: cmemc [OPTIONS] [CMEM_BASE_URI|SSL_VERIFY|REQUESTS_CA_BUNDLE|DP_API_ENDPOINT|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: `curl -H "Authorization: Bearer $(cmemc -c my admin token)" $(cmemc -c my config get DP_API_ENDPOINT)/api/custom/slug`

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

Options:

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

4.6.4 Command: config eval

Usage: cmemc [OPTIONS]

Export all configuration values of a configuration for evaluation.

The output of this command is suitable to be used by a shells `eval` command. It will output the complete configuration as `export key="value"` statements. This allows for preparation of a shell environment.

Example: `eval $(cmemc -c my config eval)`

Warning: Please be aware that credential details are shown in cleartext with this command.

Options:

- `--unset` Instead of export all configuration keys, this option will unset all key.
- `-h, --help` Show this message and exit.

4.7 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 a `DATASET_ID` (e.g: ``my-project:my-dataset``).

Note: To get a list of existing datasets, execute the ``dataset list`` command or use tab-completion.

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 metadata of a dataset.
- `list` List available datasets.
- `open` Open datasets in the browser.
- `resource` List, inspect or delete dataset file resources.

upload Upload a resource file to a dataset.

4.7.1 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:

--filter <TEXT TEXT>... List datasets based on meta data. First parameter **--filter CHOICE** can be one of ['project', 'regex', 'tag', 'type']. The second parameter is based on CHOICE.

--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.

4.7.2 Command: dataset delete

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

Delete datasets.

This command deletes existing datasets in integration projects from Corporate Memory. The corresponding dataset resources will not be deleted.

Warning: Datasets will be deleted without prompting.

Note: Datasets can be listed by using the `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.

--filter <TEXT TEXT>... Delete datasets based on meta data. First parameter
                        --filter CHOICE can be one of ['project', 'regex',
                        'tag', 'type']. The second parameter is based on
                        CHOICE.

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

4.7.3 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.

Note: Datasets can be listed by using the `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.
```

4.7.4 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.

Note: Datasets can be listed by using the ``dataset list`` command.

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

Options:

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

4.7.5 Command: dataset inspect

Usage: `cmemc [OPTIONS] DATASET_ID`

Display metadata of a dataset.

Note: Datasets can be listed by using the ``dataset list`` command.

Options:

`--raw` Outputs raw JSON.

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

4.7.6 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:

<code>-t, --type TEXT</code>	The dataset type of the dataset to create. Example types are 'csv','json' and 'eccencaDataPlatform' (-> Knowledge Graph).
<code>--project TEXT</code>	The project, where you want to create the dataset in. If there is only one project in the workspace, this option can be omitted.
<code>-p, --parameter <TEXT TEXT>...</code>	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.
<code>--replace</code>	Replace remote file resources in case there already exists a file with the same name.
<code>--id TEXT</code>	The dataset ID of the dataset to create. The dataset ID will be automatically created in case it is not present.
<code>--help-types</code>	Lists all possible dataset types on given Corporate Memory instance. Note that this option already needs access to the instance.
<code>--help-parameter</code>	Lists all possible (optional and mandatory) parameter for a dataset type. Note that this option already needs access to the instance.
<code>-h, --help</code>	Show this message and exit.

4.7.7 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.

4.8 Command group: dataset resource

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

List, inspect or delete dataset file resources.

File resources are identified by its name and project ID.

Options:

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

Commands:

delete Delete file resources.
inspect Display all meta data of a file resource.
list List available file resources.
usage Display all usage data of a file resource.

4.8.1 Command: dataset resource list

Usage: cmemc [OPTIONS]

List available file resources.

Outputs a table or a list of dataset resources (files).

Options:

--raw Outputs raw JSON.
--id-only Lists only resource names and no other meta data.
This is useful for piping the IDs into other commands.
--filter <TEXT TEXT>... Filter file resources based on a meta data. First parameter CHOICE can be one of ['project', 'regex']. The second parameter is based on CHOICE, e.g. a project ID or a regular expression string.
-h, --help Show this message and exit.

4.8.2 Command: dataset resource delete

Usage: cmemc [OPTIONS] [RESOURCE_IDS]...

Delete file resources.

You have three selection mechanisms: with specific IDs, you will delete only these resources; by using `--filter` you will delete resources based on the filter type and value; by using `--all` will delete all resources.

Options:

<code>--force</code>	Delete resource even if in use by a task.
<code>-a, --all</code>	Delete all resources. This is a dangerous option, so use it with care.
<code>--filter <TEXT TEXT>...</code>	Filter file resources based on a meta data. First parameter CHOICE can be one of ['project', 'regex']. The second parameter is based on CHOICE, e.g. a project ID or a regular expression string.
<code>-h, --help</code>	Show this message and exit.

4.8.3 Command: dataset resource inspect

Usage: cmemc [OPTIONS] RESOURCE_ID

Display all meta data of a file resource.

Options:

<code>--raw</code>	Outputs raw JSON.
<code>-h, --help</code>	Show this message and exit.

4.8.4 Command: dataset resource usage

Usage: cmemc [OPTIONS] RESOURCE_ID

Display all usage data of a file resource.

Options:

<code>--raw</code>	Outputs raw JSON.
<code>-h, --help</code>	Show this message and exit.

4.9 Command group: graph

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

List, import, export, delete, count, tree or open graphs.

Graphs are identified by an IRI.

Note: The get a list of existing graphs, execute the `graph 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.

4.9.1 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.

4.9.2 Command: graph tree

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

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

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

Imported graphs which do not exist are shown as ``[missing: IRI]``. Imported graphs which will result in an import cycle are shown as ``[ignored: IRI]``. Each graph is shown with label and IRI.

Options:

- `-a, --all` Show tree of all (readable) graphs.
- `--raw` Outputs raw JSON of the graph importTree 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. The output with this option is a sorted, flat, de-duplicated list of existing graphs.
- `-h, --help` Show this message and exit.

4.9.3 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.

4.9.4 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:

<code>-a, --all</code>	Export all readable graphs.
<code>--include-imports</code>	Export selected graph(s) and all graphs which are imported from these selected graph(s).
<code>--create-catalog</code>	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.
<code>--output-dir DIRECTORY</code>	Export to this directory.
<code>--output-file FILE</code>	Export to this file. [default: -]
<code>-t, --filename-template TEXT</code>	Template for the export file name(s). Used together with --output-dir. Possible placeholders are (Jinja2): <code>{{hash}}</code> - sha256 hash of the graph IRI, <code>{{iriname}}</code> - graph IRI converted to filename, <code>{{connection}}</code> - from the --connection option and <code>{{date}}</code> - the current date as YYYY-MM-DD. The file suffix will be appended. Needed directories will be created. [default: <code>{{hash}}</code>]
<code>--mime-type [application/n-triples text/turtle]</code>	Define the requested mime type [default: application/n-triples]
<code>-h, --help</code>	Show this message and exit.

4.9.5 Command: graph delete

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

Delete graph(s) from the store.

Options:

<code>-a, --all</code>	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.
```

4.9.6 Command: graph import

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

Import graph(s) to the store.

If input is a 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 the graph - instead of just adding new  
                   triples the graph.
```

```
--skip-existing    Skip importing a file if the target graph already exists in  
                   the store. Note that the graph list is fetched once at the  
                   beginning of the process, so that you can still add  
                   multiple files to one single graph (if it does not exist).
```

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

4.9.7 Command: graph open

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

Open / explore a graph in the browser.

Options:

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

4.10 Command group: project

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

List, import, export, create, delete or open projects.

Projects are identified by a PROJECT_ID.

Note: To get a list of existing projects, execute the `project list` command or use tab-completion.

Options:

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

Commands:

create Create projects.
delete Delete projects.
export Export projects to files.
import Import a project from a file or directory.
list List available projects.
open Open projects in the browser.
reload Reload projects from the workspace provider.

4.10.1 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.

4.10.2 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.

4.10.3 Command: project export

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

Export projects to files.

Projects can be exported with different export formats. The default type is a zip archive which includes metadata 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).

Note: Projects can be listed by using the ``project list`` command.

You can use the template string to create subdirectories.

Example: `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.
```

4.10.4 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.
```

4.10.5 Command: project delete

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

Delete projects.

This deletes existing data integration projects from Corporate Memory.

Warning: Projects will be deleted without prompting!

Note: Projects can be listed by using the `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.

4.10.6 Command: project create

Usage: cmemc [OPTIONS] PROJECT_IDS...

Create projects.

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

Note: Projects can be listed by using the `project list` command.

Options:

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

4.10.7 Command: project reload

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

Reload projects from the workspace provider.

This command reloads all tasks of a project from the workspace provider. This is the same as the `workspace reload` command, but for a single project only.

Note: You need this in case you changed project data externally or loaded a project which uses plugins which are not installed yet. In this case, install the plugin(s) and reload the project afterwards.

Warning: Depending on the size your datasets esp. your Knowledge Graphs, reloading a project can take a long time to re-create the path caches.

Options:

```
-a, --all    Reload all projects
-h, --help   Show this message and exit.
```

4.11 Command group: query

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

List, execute, get status or open SPARQL queries.

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

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).

Note: In order to get a list of queries from the query catalog, execute the `query list` command or use tab-completion.

Options:

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

Commands:

```
cancel    Cancel a running query.
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.
replay     Re-execute queries from a replay file.
status     Get status information of executed and running queries.
```

4.11.1 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, a 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:

<code>--accept TEXT</code>	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]
<code>--no-imports</code>	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.
<code>--base64</code>	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.
<code>-p, --parameter <TEXT TEXT>...</code>	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.
<code>--limit INTEGER</code>	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.

<code>--offset INTEGER</code>	Override or set the OFFSET in the executed SELECT query.
<code>--distinct</code>	Override the SELECT query by make the result set DISTINCT.
<code>--timeout INTEGER</code>	Set max execution time for query evaluation (in milliseconds).
<code>-h, --help</code>	Show this message and exit.

4.11.2 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.

4.11.3 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.

4.11.4 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 ID as a parameter.

Options:

- | | |
|--|--|
| <code>--id-only</code> | Lists only query identifier and no labels or other meta data. This is useful for piping the ids into other cmemc commands. |
| <code>--raw</code> | Outputs raw JSON response of the query status API. |
| <code>--filter <TEXT TEXT>...</code> | Filter queries based on execution status and time. First parameter <code>--filter CHOICE</code> can be one of ['graph', 'regex', 'slower-than', 'status', 'trace-id', 'type', 'user']. The second parameter is based on CHOICE, e.g. int in case of slower-than, or a regular expression string. |
| <code>-h, --help</code> | Show this message and exit. |

4.11.5 Command: query replay

Usage: `cmemc [OPTIONS] REPLAY_FILE`

Re-execute queries from a replay file.

This command reads a `REPLAY_FILE` and re-executes the logged queries. A `REPLAY_FILE` is a JSON document which is an array of JSON objects with at least a key ``queryString`` holding the query text OR a key ``iri`` holding the IRI of the query in the query catalog. It can be created with the ``query status`` command.

Example: `query status --raw > replay.json`

The output of this command shows basic query execution statistics.

The queries are executed one after another in the order given in the input `REPLAY_FILE`. Query placeholders / parameters are ignored. If a query results in an error, the duration is not counted.

The optional output file is the same JSON document which is used as input, but each query object is annotated with an additional ``replays`` object, which is an array of JSON objects which hold values for the replay|loop|run IDs, start and end time as well as duration and other data.

Options:

```
--raw           Output the execution statistic as raw JSON.
--loops INTEGER  Number of loops to run the replay file. [default: 1]
--wait INTEGER   Number of seconds to wait between query executions.
                  [default: 0]

--output-file FILE Save the optional output to this file. Input and output
                  of the command can be the same file. The output is
                  written at the end of a successful command execution.
                  The output can be stdout ('-') - in this case, the
                  execution statistic output is oppressed.

--run-label TEXT  Optional label of this replay run.
-h, --help       Show this message and exit.
```

4.11.6 Command: query cancel

Usage: `cmemc [OPTIONS] QUERY_ID`

Cancel a running query.

With this command, you can cancel a running query. Depending on the backend triple store, this will result in a broken result stream (stardog, neptune and virtuoso) or a valid result stream with incomplete results (graphdb)

Options:

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

4.12 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.

4.12.1 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.

4.12.2 Command: vocabulary list

Usage: cmemc [OPTIONS]

Output a list of vocabularies.

Vocabularies are graphs (see `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.
```

4.12.3 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 command.

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

4.12.4 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 command.

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

4.12.5 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:

<code>--namespace <TEXT TEXT>...</code>	In case the imported vocabulary file does not include a preferred namespace prefix, you can manually add a namespace prefix with this option. Example: <code>--namespace ex https://example.org/</code>
<code>--replace</code>	Replace (overwrite) existing vocabulary, if present.
<code>-h, --help</code>	Show this message and exit.

4.13 Command group: vocabulary cache

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

List und update the vocabulary cache.

Options:

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

Commands:

<code>list</code>	Output the content of the global vocabulary cache.
<code>update</code>	Reload / updates the data integration cache for a vocabulary.

4.13.1 Command: vocabulary cache update

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

Reload / updates the data integration cache for a vocabulary.

Options:

<code>-a, --all</code>	Update cache for all installed vocabularies.
<code>-h, --help</code>	Show this message and exit.

4.13.2 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.

4.14 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 a 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.
- open Open a workflow in your browser.
- scheduler List, inspect, enable/disable or open scheduler.
- status Get status information of workflow(s).

4.14.1 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:

<code>-a, --all</code>	Execute all available workflows.
<code>--wait</code>	Wait until all executed workflows are completed.
<code>--polling-interval INTEGER RANGE</code>	How many seconds to wait between status polls. Status polls are cheap, so a higher polling interval is most likely not needed. [default: 1]
<code>-h, --help</code>	Show this message and exit.

4.14.2 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:

<code>-i, --input FILE</code>	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.
-------------------------------	--

<code>-o, --output FILE</code>	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.
<code>--input-mimetype [guess application/xml application/json text/csv]</code>	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
<code>--output-mimetype [guess application/xml application/json application/n-triples application/vnd.openxmlformats-officedocument.wordprocessingml.document]</code>	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).
<code>--autoconfig / --no-autoconfig</code>	Setup auto configuration of input datasets, e.g. in order to process CSV files with semicolon- instead of comma-separation. [default: True]
<code>-h, --help</code>	Show this message and exit.

4.14.3 Command: workflow list

Usage: cmemc [OPTIONS]

List available workflow.

Options:

- `--filter <TEXT TEXT>...` List workflows based on meta data. First parameter `--filter CHOICE` can be one of ['io', 'project', 'regex', 'tag']. The second parameter is based on CHOICE.
- `--id-only` Lists only workflow identifier and no labels or other meta data. This is useful for piping the IDs

```
into other commands.

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

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

4.14.4 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.
```

4.14.5 Command: workflow open

```
Usage: cmemc [OPTIONS] WORKFLOW_ID

Open a workflow in your browser.

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

4.15 Command group: workflow scheduler

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

List, inspect, enable/disable or open scheduler.

Schedulers execute workflows in specified intervals. They are identified
with a SCHEDULER_ID. To get a list of existing schedulers, execute the
list command or use tab-completion.
```

Options:

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

Commands:

disable Disable scheduler(s).

enable Enable scheduler(s).

inspect Display all meta data of a scheduler.

list List available scheduler.

open Open scheduler(s) in the browser.

4.15.1 Command: workflow scheduler open

Usage: cmemc [OPTIONS] SCHEDULER_IDS...

Open scheduler(s) in the browser.

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

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

Options:

--workflow Instead of opening the scheduler page, open the page of the scheduled workflow.

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

4.15.2 Command: workflow scheduler list

Usage: cmemc [OPTIONS]

List available scheduler.

Outputs a table or a list of scheduler IDs which can be used as reference for the scheduler commands.

Options:

--raw Outputs raw JSON.

--id-only Lists only task 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.
```

4.15.3 Command: workflow scheduler inspect

```
Usage: cmemc [OPTIONS] SCHEDULER_ID
```

Display all meta data of a scheduler.

Options:

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

4.15.4 Command: workflow scheduler disable

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

Disable scheduler(s).

The command accepts multiple scheduler IDs which results in disabling them one after the other.

Options:

```
-a, --all  Disable all scheduler.  
-h, --help Show this message and exit.
```

4.15.5 Command: workflow scheduler enable

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

Enable scheduler(s).

The command accepts multiple scheduler IDs which results in enabling them one after the other.

Options:

```
-a, --all  Enable all scheduler.  
-h, --help Show this message and exit.
```