



# Corporate Memory Control (cmemc)

v22.1.1

Manual



# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	About eccenca Corporate Memory Control (cmemc)	5
1.2	Scope of delivery	6
<b>2</b>	<b>Installation</b>	<b>7</b>
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
<b>3</b>	<b>Configuration</b>	<b>9</b>
<b>4</b>	<b>Reference</b>	<b>12</b>
4.1	Command group: admin	12
4.1.1	Command: admin showcase	12
4.1.2	Command: admin bootstrap	13
4.1.3	Command: admin status	13
4.1.4	Command: admin token	14
4.2	Command group: admin metrics	14
4.2.1	Command: admin metrics get	15
4.2.2	Command: admin metrics inspect	16
4.2.3	Command: admin metrics list	16
4.3	Command group: admin workspace	17
4.3.1	Command: admin workspace export	17
4.3.2	Command: admin workspace import	18
4.3.3	Command: admin workspace reload	18
4.4	Command group: admin workspace python	18
4.4.1	Command: admin workspace python install	19
4.4.2	Command: admin workspace python uninstall	19
4.4.3	Command: admin workspace python list	19
4.4.4	Command: admin workspace python list-plugins	20
4.5	Command group: admin store	20
4.5.1	Command: admin store showcase	21
4.5.2	Command: admin store bootstrap	21
4.5.3	Command: admin store export	22
4.5.4	Command: admin store import	22

4.6	Command group: config . . . . .	22
4.6.1	Command: config list . . . . .	24
4.6.2	Command: config edit . . . . .	24
4.6.3	Command: config get . . . . .	25
4.6.4	Command: config eval . . . . .	25
4.7	Command group: dataset . . . . .	26
4.7.1	Command: dataset list . . . . .	26
4.7.2	Command: dataset delete . . . . .	27
4.7.3	Command: dataset download . . . . .	27
4.7.4	Command: dataset upload . . . . .	28
4.7.5	Command: dataset inspect . . . . .	28
4.7.6	Command: dataset create . . . . .	29
4.7.7	Command: dataset open . . . . .	30
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 . . . . .	32
4.9	Command group: graph . . . . .	32
4.9.1	Command: graph count . . . . .	32
4.9.2	Command: graph tree . . . . .	33
4.9.3	Command: graph list . . . . .	33
4.9.4	Command: graph export . . . . .	34
4.9.5	Command: graph delete . . . . .	35
4.9.6	Command: graph import . . . . .	35
4.9.7	Command: graph open . . . . .	36
4.10	Command group: project . . . . .	36
4.10.1	Command: project open . . . . .	36
4.10.2	Command: project list . . . . .	37
4.10.3	Command: project export . . . . .	37
4.10.4	Command: project import . . . . .	38
4.10.5	Command: project delete . . . . .	39
4.10.6	Command: project create . . . . .	39
4.11	Command group: query . . . . .	39
4.11.1	Command: query execute . . . . .	40
4.11.2	Command: query list . . . . .	41
4.11.3	Command: query open . . . . .	42
4.11.4	Command: query status . . . . .	42
4.11.5	Command: query replay . . . . .	43
4.12	Command group: vocabulary . . . . .	44
4.12.1	Command: vocabulary open . . . . .	44
4.12.2	Command: vocabulary list . . . . .	45
4.12.3	Command: vocabulary install . . . . .	45
4.12.4	Command: vocabulary uninstall . . . . .	45

4.12.5 Command: vocabulary import . . . . .	46
4.13 Command group: vocabulary cache . . . . .	46
4.13.1 Command: vocabulary cache update . . . . .	47
4.13.2 Command: vocabulary cache list . . . . .	47
4.14 Command group: workflow . . . . .	47
4.14.1 Command: workflow execute . . . . .	48
4.14.2 Command: workflow io . . . . .	48
4.14.3 Command: workflow list . . . . .	49
4.14.4 Command: workflow status . . . . .	50
4.14.5 Command: workflow open . . . . .	50
4.15 Command group: workflow scheduler . . . . .	51
4.15.1 Command: workflow scheduler open . . . . .	51
4.15.2 Command: workflow scheduler list . . . . .	51
4.15.3 Command: workflow scheduler inspect . . . . .	52
4.15.4 Command: workflow scheduler disable . . . . .	52
4.15.5 Command: workflow scheduler enable . . . . .	52

# 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 package<sup>1</sup>, so installation can be done with pip or pipx<sup>2</sup>:

```
$ 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
```

---

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

<sup>2</sup><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](#)<sup>3</sup>.

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.

---

<sup>3</sup><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)<sup>1</sup>. 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-`

<sup>1</sup><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 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.  
metrics List and get metrics.  
showcase Create showcase data.  
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 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. If the version information can not be retrieved, UNKNOWN is shown if the endpoint is not available or ERROR is shown, if the endpoints returns an error.

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 then your DataPlatform component.

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

```
cmemc config list | parallel --ctag 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.

--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/](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:

--job [DP]                      The job from which the metrics data is fetched.  
                                 [default: DP]

--filter <TEXT TEXT>...        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.

```
--enforce-table    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.

--raw              Outputs raw prometheus sample classes.

-h, --help         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:

```
--job [DP]  The job from which the metrics data is fetched. [default: DP]
--raw       Outputs raw JSON of the table data.
-h, --help  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:

```
-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.
```

### 4.3.2 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.
```

### 4.3.3 Command: admin workspace reload

```
Usage: cmemc [OPTIONS]

Reload the workspace from the backend.

Options:
  -h, --help  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:
```

<code>install</code>	Install a python package to the workspace.
<code>list</code>	List installed python packages.
<code>list-plugins</code>	List installed workspace plugins.
<code>uninstall</code>	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 that the plugin discovery is limited to specific packages.

Options:

- `--raw` Outputs raw JSON.
- `--id-only` Lists only plugin 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:

```
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.

#### 4.6.2 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_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.

### 4.6.4 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.

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

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 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.
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:

--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.

#### 4.7.2 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.

#### 4.7.3 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 (eccencaDataplatfrom) 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.

#### 4.7.4 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.

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.

#### 4.7.5 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.

#### 4.7.6 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 &lt;TEXT TEXT&gt;...</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.

```
-h, --help          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:

- |  |   |
|--|---|
| <code>--raw</code>                         | Outputs raw JSON.   |
| <code>--id-only</code>                     | Lists only resource names and no other meta data.<br>This is useful for piping the IDs into other commands.   |
| <code>--filter &lt;TEXT TEXT&gt;...</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.   |

#### 4.8.2 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 &lt;TEXT TEXT&gt;...</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:**

--raw           Outputs raw JSON.  
-h, --help   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:**

--raw           Outputs raw JSON.  
-h, --help   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. 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.

#### 4.9.1 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): {{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.
```

#### 4.9.5 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.
```

#### 4.9.6 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 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 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.

### 4.10.1 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 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.
<code>--output-dir DIRECTORY</code>	The base directory, where the project files will be created. If this directory does not exist, it will be silently created. [default: .]
<code>--type TEXT</code>	Type of the exported project file(s). Use the <code>--help-types</code> option or tab completion to see a list of possible types. [default: xmlZip]
<code>-t, --filename-template TEXT</code>	Template for the export file name(s). Possible placeholders are (Jinja2): <code>{{id}}</code> (the project ID), <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}}-{{id}}.project</code> ]
<code>--extract</code>	Export projects to a directory structure instead of a ZIP archive. Note that the <code>--filename-template</code> 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.
<code>--help-types</code>	Lists all possible export types.
<code>-h, --help</code>	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 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.

#### 4.10.6 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.

### 4.11 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).

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.  
`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:

`--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.

#### 4.11.2 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 &lt;CHOICE TEXT&gt;...</code>	Filter queries based on execution status and time. First parameter CHOICE can be 'slower-than', 'status' or 'type'. The second parameter has to be a finished or running, in case of the 'status' filter, a time in milliseconds in case of the 'slower-than' filter or a query type in case of the 'type' filter.
<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, e.g. ``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.
```

## 4.12 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 '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.

### 4.12.3 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 --filter installable" 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 --filter installed" 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:

- replace Replace (overwrite) existing vocabulary, if present.
- h, --help 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:

- list Output the content of the global vocabulary cache.

update 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:

- a, --all Update cache for all installed vocabularies.
- h, --help 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 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.
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:

- 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-officedocument.wordprocessingml.document]      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.

### 4.14.3 Command: workflow list

Usage: cmemc [OPTIONS]

List available workflow ids.

**Options:**

- |  |   |
|--|---|
| <code>--raw</code>                           | Outputs raw JSON objects of workflow task search API response.  |
| <code>--id-only</code>                       | Lists only workflow identifier and no labels or other meta data. This is useful for piping the IDs into other commands.   |
| <code>--filter &lt;CHOICE TEXT&gt;...</code> | Filter workflows based on project or suitability for the <code>io</code> command. First parameter <code>CHOICE</code> 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. |
| <code>-h, --help</code>                      | Show this message and exit.   |

#### 4.14.4 Command: workflow status

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

Get status information of workflow(s).

**Options:**

- |  |  |
|--|--|
| <code>--project TEXT</code>  | The project, from which you want to list the workflows. Project IDs can be listed with the 'project list' command. |
| <code>--raw</code>   | Output raw JSON info.  |
| <code>--filter [Idle Not executed Finished Cancelled Failed Successful Canceling Running Waiting]</code> | Show only workflows of a specific status.  |
| <code>-h, --help</code>  | Show this message and exit.  |

#### 4.14.5 Command: workflow open

Usage: `cmemc [OPTIONS] WORKFLOW_ID`

Open a workflow in your browser.

**Options:**

- |                         |                             |
|-------------------------|-----------------------------|
| <code>-h, --help</code> | 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 SCHEDULERID. 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.