A Softer Space DE, Inhaber Tom Moya Schiller | Heerstraße 3 b | 60488 Frankfurt am Main, Germany

Latest version of this document:
asofterspace.com/cdm/manual.pdf

Phone: +49 176 511 27307
Email: info@asofterspace.com
Website: asofterspace.com
VAT Nr: DE319451065

Documentation for:
cdm commandline tool
version 0.0.1.3beta(25)

CDM MANUAL

Date: 8th October 2018

The cdm commandline tool enables quick and simple interactions with EGS-CC CDM files directly from the commandline, both under Linux and Windows.

These interactions can involve queries, such as finding elements based on their ID or name, printing the MCM tree, and validating the correctness of the CDM, as well as actual modifications, such as setting the name of the root element or converting the CDM to a different version.

Contents

Default Setup

To set up the cdm commandline tool you can just download a zip file, call a setup script, and it should start working.

In particular, you can download the latest zip from:

```
http://asofterspace.com/cdm/cdm.zip
```

To run the cdm commandline tool on your computer, you need to install Java on it.
(Any JRE or JVM of Java 7 or higher will be sufficient; for using very large CDM files, a 64-Bit version will be necessary.)

To check that you have Java installed, you can type in a terminal:

```
java -version
```

Once Java is found, unzip the cdm commandline tool file, e.g. under Linux by typing:

```
unzip cdm.zip
```

Finally, to register the cdm command with your terminal, call under Windows:

```
cd cdm\windows
install_from_zip.bat
```

Or under Linux:

```
cd cdm/linux
./install_from_zip.sh
```

The `cdm` command will now be available in the current terminal session.
Especially under Windows, it might be necessary to restart the machine before it also becomes available in other terminal sessions.

Setup using Manual Build

If you want to build the cdm commandline tool yourself, then you first of all should ensure that you have a JDK (Java Development Kit) of Java 7 or higher on your machine. Ideally, you will be using a 64-Bit version of Java, as this enables using a larger heap size; a 32-Bit version will be enough for small test CDMs and has on occasion been sufficient for loading real mission-size CDMs, but might not in all cases be.
You also have to ensure that `javac` is on the `PATH` and can be found.
To check both, you can type in a terminal:

```
java -version and javac -version
```

Once you have set up the JDK, clone the cdm repository onto your machine:

```
git clone https://github.com/ASofterSpace/cdm.git
```

Finally, build the latest version of the cdm commandline tool by calling under Windows:

```
cd cdm\windows
install_latest.bat
```

Or under Linux:

```
cd cdm/linux
./install_latest.sh
```

The `cdm` command will now be available in the current terminal session.
Especially under Windows, it might be necessary to restart the machine before it also becomes available in other terminal sessions.

Should you want to create a zip release file of what you just built, then you have to get your hands on a Java 7 `rt.jar` (such that the released zip can be created in a way compatible with Java 7 and above.)
Once you have it, put this file at the location:

```
cdm/other/java7_rt.jar
```

Then, under Linux you can call:

```
./release.sh
```

(from the same directory in which you also executed the `./install_latest.sh` command.)

Basic Operations

To get a list of all available commands, you can type:

```
cdm help
```

If you are interested in one command in particular, you can also enter:

```
cdm help <command>
```

Most commands expect to work on a particular CDM; if you already have a CDM that you are interested in working with, then you can use that one – if not, then you can also create a small test CDM directly from the commandline.

The following command will achieve this for you:

```
cdm create newcdm
```

where `newcdm` is the path under which the new CDM is supposed to be stored; in this case, it is a new folder called `newcdm` inside the current working directory.

Optionally, you can also specify a target CDM version and a template that should be used, such as:

```
cdm create -v 1.13.0bd1 -t root_route_sap newcdm_rs
```

Now a new CDM is being created in the folder `newcdm_rs` which contains a root element, a default route and a default service access point, as these belong to the template `root_route_sap`.

We can now get some information about the two newly created CDMs. To do so, call:

```
cdm info newcdm
```

This should produce output such as:

```
CDM version: 1.14.0
CDM version prefix: http://www.esa.int/egscc/
CDM compatible with EGS-CC release: IR4
CDM compatible with RTF Framework CDM editor version: 0.18.7
```

However, for the second CDM we specified a different version when we created it.

You can satisfy yourself that it truly has been created with a different version by calling:

```
cdm info newcdm_rs
```

Let's convert the second CDM to the same version as the first one by calling:

```
cdm convert -v 1.14.0 newcdm_rs
```

The second CDM has now been converted in-place to version 1.14.0.

We can also use the destination argument `-d` to specify a different destination for the conversion result, such as:

```
cdm convert -v 1.12.1 -d newcdm_rs_2 newcdm_rs
```

Here, `newcdm_rs` was the CDM that was read and converted, with the conversion result being stored into the destination `newcdm_rs_2`.

We can compare the CDMs before and after conversion:

```
cdm compare newcdm_rs newcdm_rs_2
```

This shows us that even for a tiny test CDM, already quite a few changes had to be performed during the conversion.

However, how tiny is this test CDM actually?
Let's look at the CDM tree:

```
cdm tree newcdm_rs
```

This shows us just the monitoring and control element `mcmRoot` and nothing more.
If we are interested in getting more information about the root element, we can type:

```
cdm root newcdm_rs
```

However, we can use the `cdm root` command also to assign a new name to the root element. This is especially helpful when trying to load test CDMs with different root names in the same EGS-CC-based system instance, as we can simply change the CDM root names instead of having to re-configure the system to accept a different root every time before changing the CDM:

```
cdm root -n newName newcdm_rs
```

We can also search for a command with a particular name with the `cdm find` command:

```
cdm find -n newName newcdm_rs
```

Finally, we can validate that all our playing around did not create problems with the CDM itself:

```
cdm validate newcdm_rs
```

(This validation is intended to augment the already existing validation in EGS-CC – that is, it will most likely not flag the same errors, but might find errors that EGS-CC would not report.)

License and Support

We at A Softer Space really love the Unlicense, which pretty much allows anyone to do anything with this source code. For more info, see the file UNLICENSE in the git repository. If you desperately need to use this source code under a different license, contact moya@asofterspace.com - I am sure we can figure something out.

Finally, if you like this tool and would be happy for it to be developed further, for bugs that you find to be fixed, and even for new features that you have in mind to be added, then just let us know and we will offer you a service contract at such great conditions that you cannot possibly say no to it. :)

In the right light, at the right time,
everything is extraordinary.
~ Aaron Rose