Workshop on CMIP6 Post-processing

Martin Schupfner (schupfner@dkrz.de),
Fabian Wachsmann (wachsmann@dkrz.de),
DKRZ

Presentation available here:
https://c6dreq.dkrz.de/info/workshop_dicad_tools.php

Oct 9th and 10th 2019

Agenda - Coarse Overview

Day 1 - 9th October:
Introduction and best practice: Processing of CMIP6-DECK experiments incl. minor customizations

Day 2 - 10th October:
Customized Post-processing: Configuration and composition of the personal post processing workflow

Agenda

9th October 2019

10:30	*Coffee reception*
11:00	**CMIP6 data in ESGF - recent status** DICAD contribution
11:15	**CMIP6 Post-processing workflow** Motivation & modularity
12:00	**CMIP6 Post-processing workflow** Reenaction of the workflow for a selected experiment
12:30	*Lunch break (nothing centrally organized, but mensa is nearby)*
13:15	**CMIP6 Post-processing workflow** Reenaction of the workflow for a selected experiment *Part II*
15:00	*Coffee break*
15:20	**CMIP6 Post-processing workflow** Reenaction of the workflow for a selected experiment *Part III*
16:30	**CMIP6 Post-processing workflow** What to keep in mind, Q&A
17:00	*End*

Workshop on CMIP6 Post-processing

- DICAD data published in ESGF - recent status

Martin Schupfner (schupfner@dkrz.de),
Fabian Wachsmann (wachsmann@dkrz.de),
DKRZ

CMIP definitions

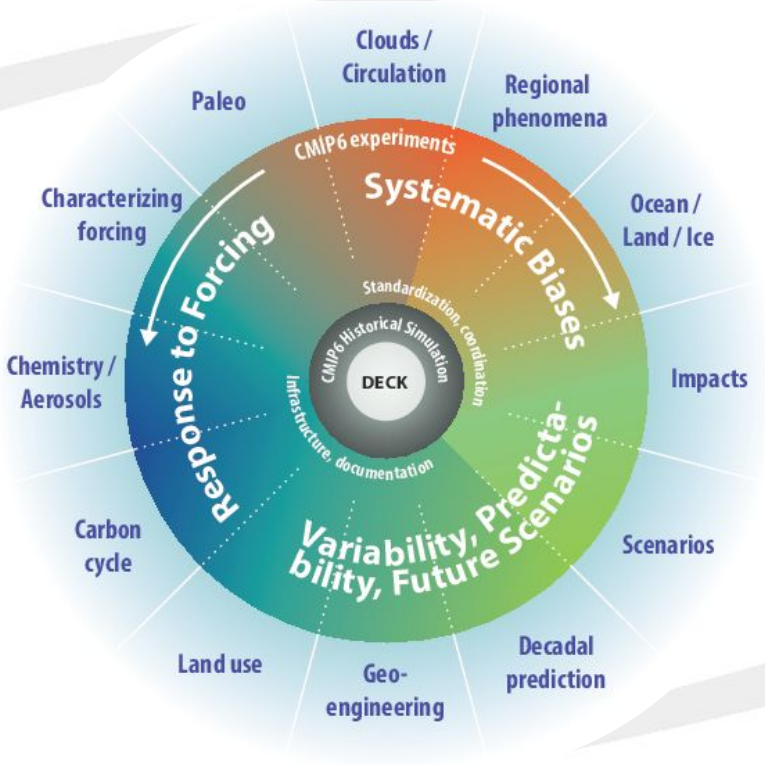
The Coupled Model Intercomparison Project (CMIP) is a project of the World Climate Research Programme (WCRP)'s Working Group of Coupled Modelling (WGCM).

- CMIP's **central goal** is to advance scientific understanding of the Earth system.
- CMIP model simulations have also been regularly assessed as part of the **IPCC Climate Assessments Reports** and various national assessments.
- CMIP defines **common experiment protocols**, forcings and output.

CMIP6 MIPs and entry card

24 registered *activity_ids*:

AerChemMIP	HighResMIP
C4MIP	ISMIP6
CDRMIP	LS3MIP
CFMIP	LLUMIP
CMIP	OMIP
CORDEX	PAMIP
DAMIP	PMIP
DCPP	RFMIP
DynVarMIP	SIMIP
FAFMIP	*ScenarioMIP*
GMMIP	VIACSAB
GeoMIP	VolMIP



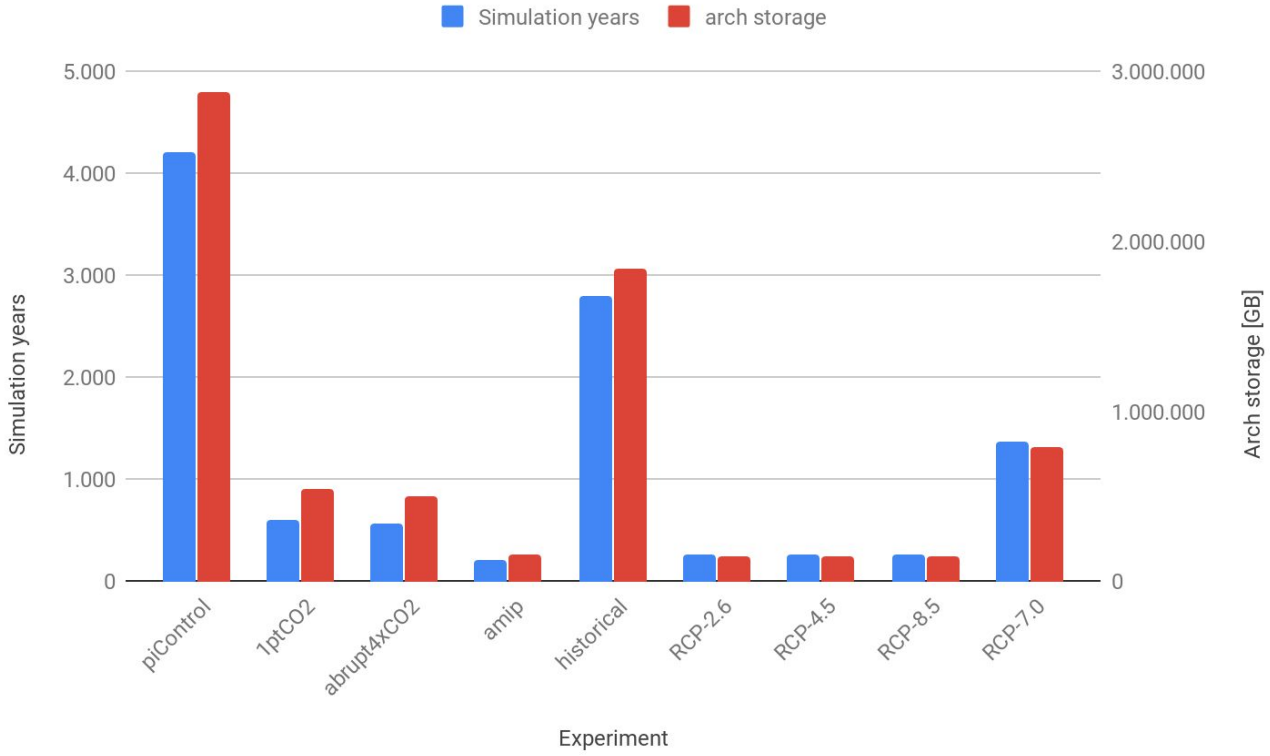
The **DECK** (Diagnosis, Evaluation and Characterization of Klima) experiments are, together with the **historical** experiment, the entry card for CMIP6 and consist of

i AMIP
ii. Pre-industrial control
iii. 1%/yr CO2
iv. Abrupt 4xCO2

CMIP includes **DECK** and **historical**

CMIP6 DICAD simulations in RZ988

Simulation years and **arch storage** for all CMIP and ScenarioMIP experiments conducted by MPI-ESM1.2.01-HR, ICON-ESM-LR, AWI-CM-1-1-MR and EMAC-2-53-AerChem within the RZ988



Total arch storage: 7Pb

CMIP5 all MPI-ESMs: 700 Tb

CMIP6 Experiments of MPI-M and DKRZ and general information

- Status of the climate model simulations within DICAD is being tracked under:
  https://redmine.dkrz.de/projects/cmip6-dicad-subproject/wiki/Climate_Model_Simulations_-_Status

- The ESGF data pool is mounted on mistral under
  /work/ik1017/CMIP6/data/CMIP6/
  and contains all CMIP6 data published and replicated in the DKRZ ESGF node
  DKRZ ESGF node for CMIP6: https://esgf-data.dkrz.de/search/cmip6-dkrz/
  DKRZ ESGF node for all projects: https://esgf-data.dkrz.de/projects/esgf-dkrz/

- Quick looks of some published variables will be available under
  https://cmip-esmvaltool.dkrz.de/ soon
  (Access soon possible with the standard DKRZ / LDAP account that is used for the mistral login.)

- Errata information about unpublished files are under
  https://errata.es-doc.org/static/index.html

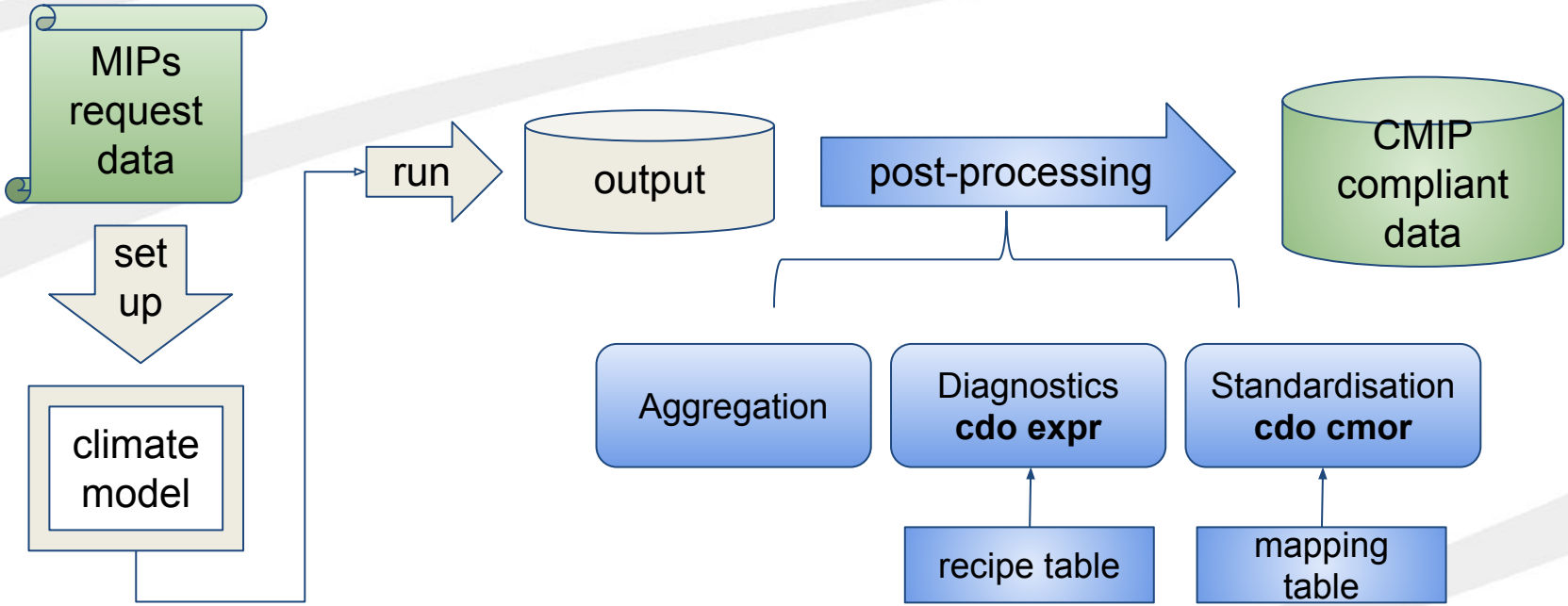
Workshop on CMIP6 Post-processing

Introduction to the DICAD Post Processing Workflow

- Motivation & workflow modularity

Martin Schupfner (schupfner@dkrz.de),
Fabian Wachsmann (wachsmann@dkrz.de),
DKRZ

CMIP6 Data Production Workflow - Post-processing definition



We are developing

- the **cdo cmor** operator
- the **c6dreq-WebGUI** to
  - create recipe/mapping tables
  - create a customized DReq
  - generate script fragments for diagnostic & standardisation

Further steps are not displayed (Monitoring, QA, Citation, Publication, Long-term archiving, …)

CMIP6 Data Standard - Why?

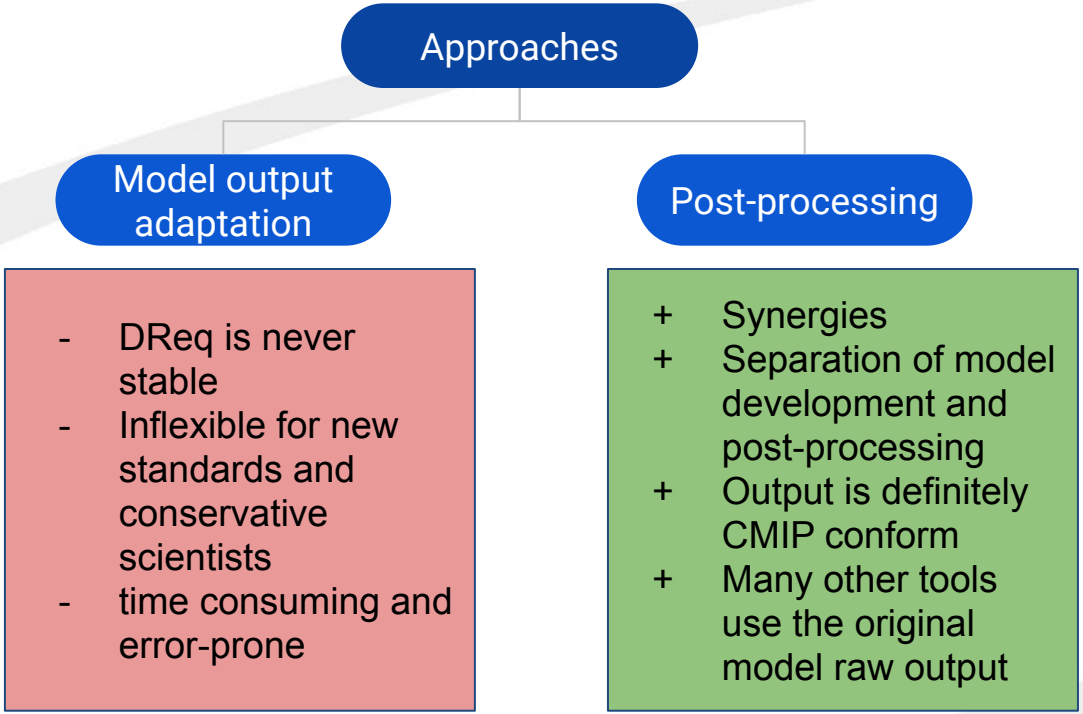
CMIP Scope
Both Scientists and their programs must be able to rely on a common data standard for **2000 unique variables** from 100 Experiments of 25 Model Intercomparison Projects (MIPs) and conducted by 100 models.

"**Systematic analysis** across models only easy to do if model output is written in

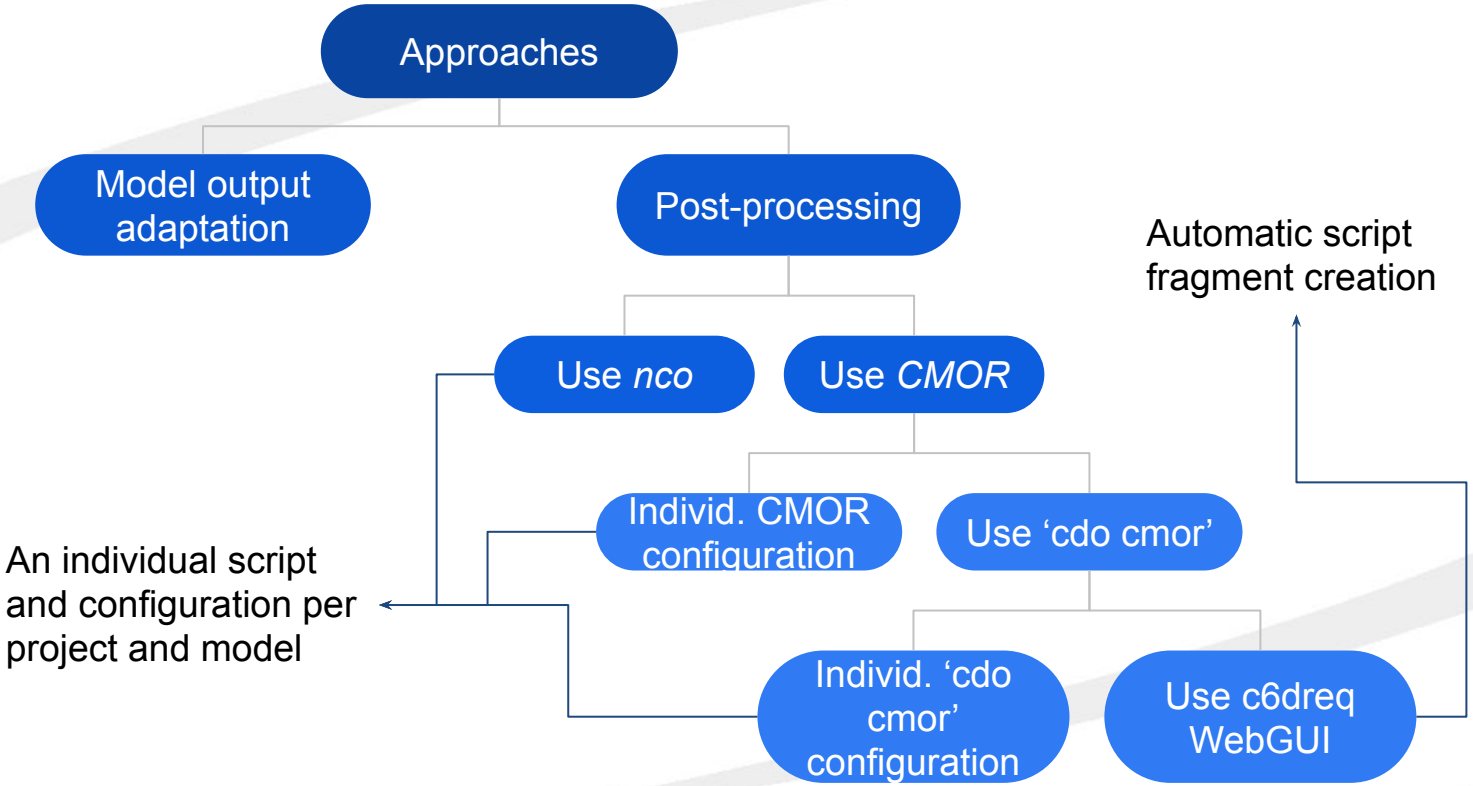
- a common format
- with files structured similarly
- and with sufficient metadata uniformly stored“

Reusability
Climate model output files should be **self descriptive**. E.g. statistical operations and interpolations over space and time are only processable if all temporal and spatial information including cells and interval bounds and a full description of the vertical axis is available in the input file.

Model output adaptation or post-processing?



CMIP6 DICAD Objectives

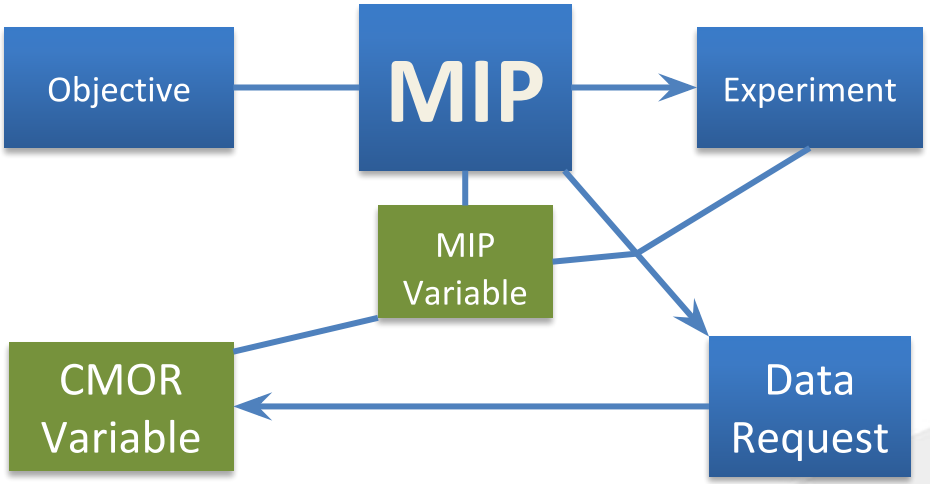


CMIP6 Dictionary

Terms	Definition
CMOR	Climate Model Output Rewriter. Software for Standardizing.
MIP-Table, CMOR-Table, CMIP-Table	JSON formatted table readable by CMOR and including many variables defined by CMIP *usually* for one specific realm and frequency.
CMOR Variable	Unique combination of Variable name and MIP-Table where it is included.
DReq / Data request	Variables requested and defined all over CMIP. Also: Software package.
Mapping / mapping table	Mapping of the raw model output on the variables defined in the CMIP standard. A mapping table contains all necessary information to clearly match a CMOR Variable with a model variable.
Infotable /cdocmorinfo	Table including keyvalues with specifications of global attributes and control keywords for (cdo) CMOR
c6, diag, pp, agg, drs	c6 = CMIP6. diag = Diagnostics. pp = Post-processing. agg = Aggregation. drs = data reference syntax.
namelist	Raw model output file naming convention.

CMIP6 Data Request - What is a “CMOR variable”?

- MIPs founded to achieve WCRP defined scientific objectives
- MIPs define Experiments, Variables and set up a data request
- CMOR-Variables are the different versions (frequency, shape, …) of a MIP-Variable



Example:

MIP-Variable: Air Temperature

CMOR Variables:

(1) **ta** - Air Temperature (zonal mean on 39 pressure levels, monthly mean)

(2) **ta** - Air Temperature (global field on model levels, monthly mean)

(3) **ta** -Air Temperature (global field on 19 pressure levels, monthly mean)

Ambiguity?

CMIP6 Data Request - What is a “CMOR variable”?

The CMOR variable name alone does not identify a CMOR variable unambiguously

→CMOR variables are organized in MIP tables

(MIP tables: tables of variables of similar frequency, realm and cell methods)

Our Example:

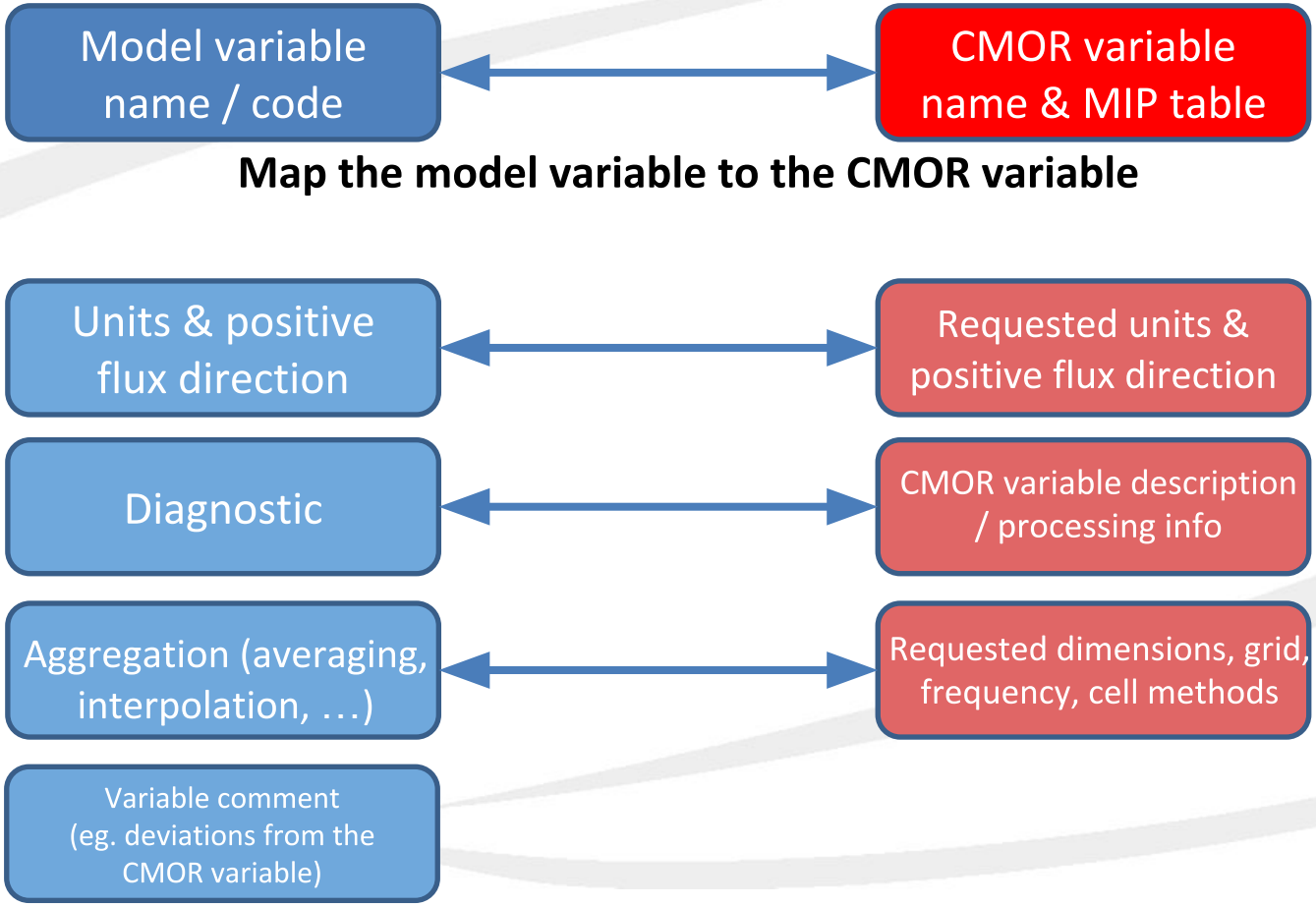
(1) **ta** - Air Temperature (zonal mean on 39 pressure levels, monthly mean) → **AERmonZ**

(2) **ta** - Air Temperature (global field on model levels, monthly mean) → **CFmon**

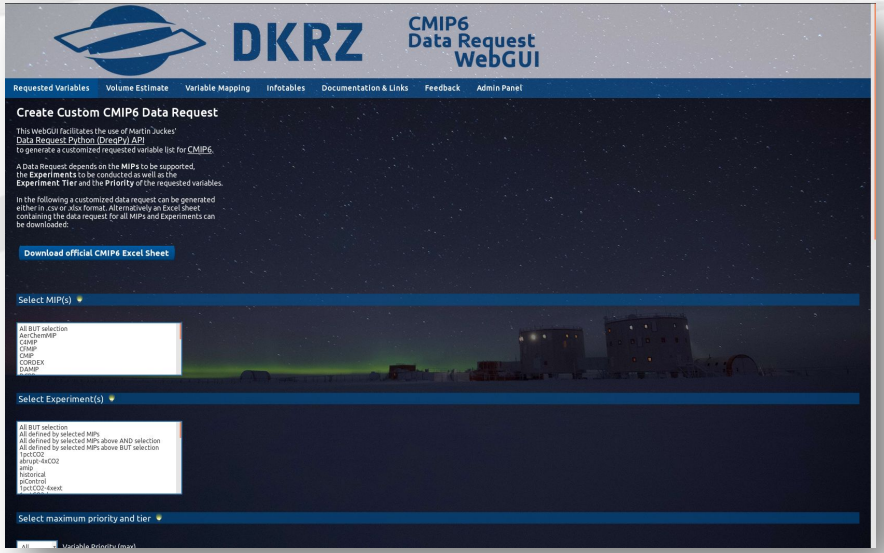
(3) **ta** -Air Temperature (global field on 19 pressure levels, monthly mean) → **Amon**

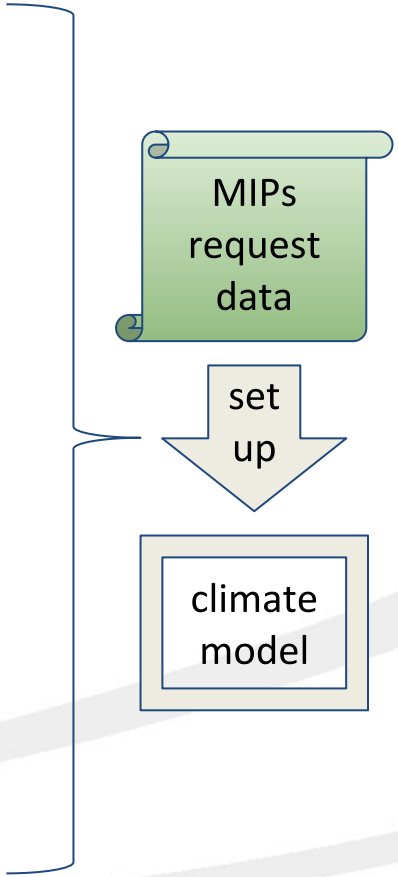
The pair “CMOR variable name” and “MIP table name” unambiguously defines the CMOR variable.

What do we understand by “mapping”?



Model Output Configuration





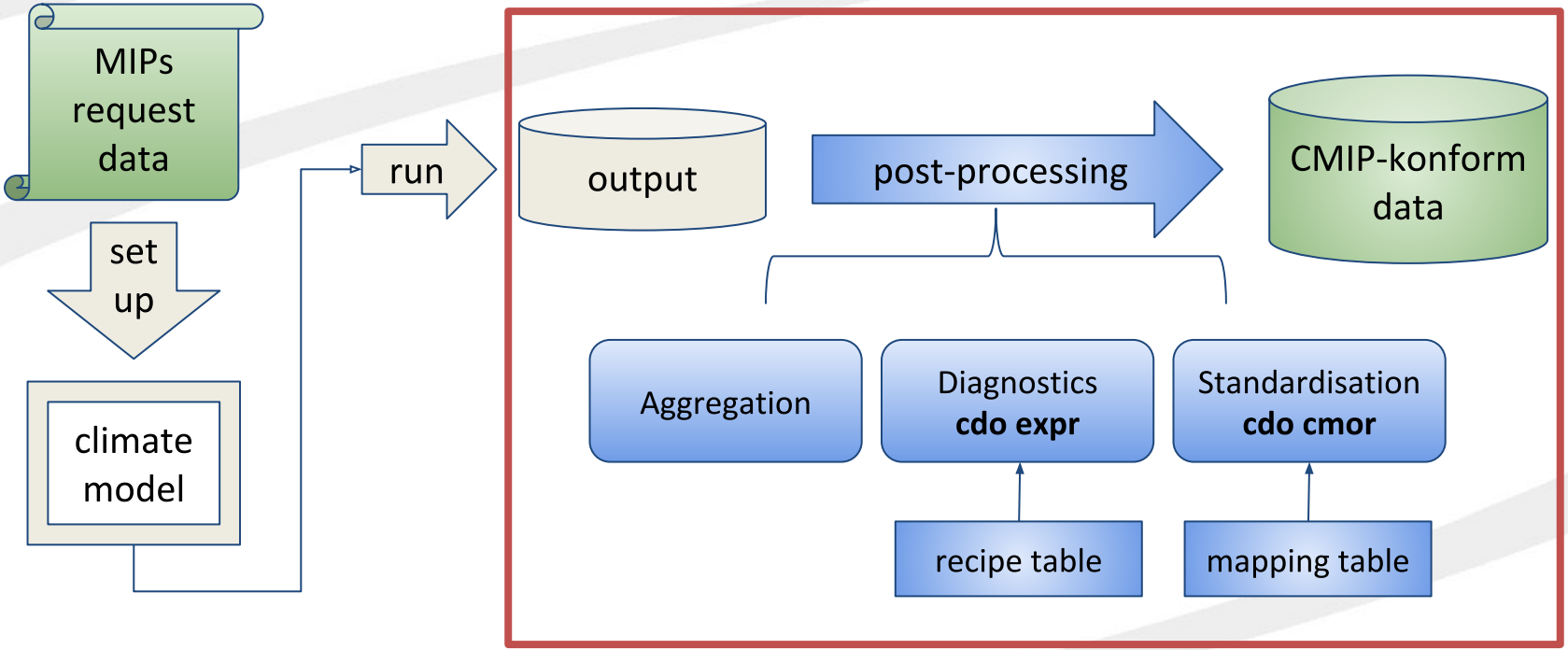
Web GUI to use the basic functions of the Data Request Python API:

- Data request as csv list
- Data request as excel document
- Volume estimate

URL: https://c6dreq.dkrz.de

HowTo: https://c6dreq.dkrz.de/info/howto2.php

CMIP6 Data Production Workflow - Modular Post-Processing



Modular post-processing - Motivation

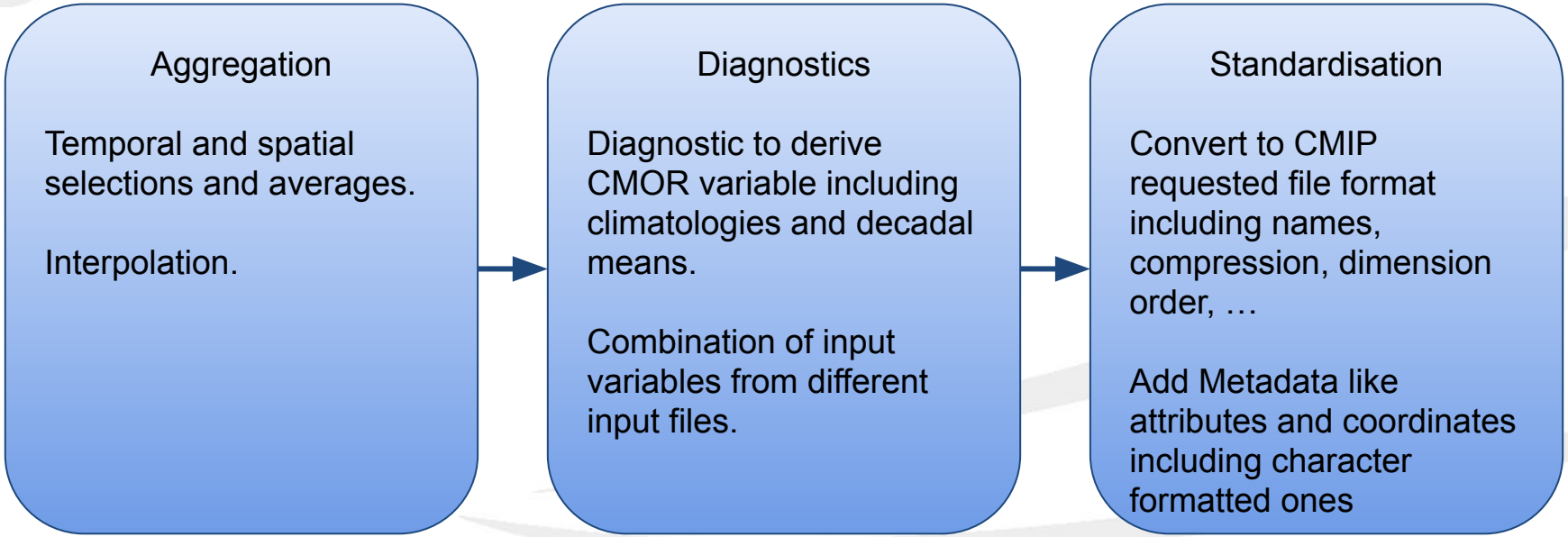
Aggregation | Diagnostics **cdo expr** | Standardisation **cdo cmor**

\+ Independent execution of each part
\+ Easier error handling and debugging

– Relatively more I/O (e.g., less usage of CDO's operator chaining)

Modular post-processing - Definitions

Aggregation | Diagnostics **cdo expr** | Standardisation **cdo cmor**



Modular post-processing - Definitions

Aggregation | Diagnostics **cdo expr** | Standardisation **cdo cmor**

Aggregation

Temporal averaging:

```
cdo monmean
day_ta.nc mon_ta.nc
```

```
cdo after ...
```

Spatial selection, interpolation and averaging:

```
cdo zonmean infile
outfile
```

Diagnostics

Technical definition: Mathematical expressions.

```
cdo expr
```

is able to interpret these.

```
cdo expr,
"pr=prc+prs" infile
outfile
```

Standardisation

Convert to CMIP requested file format including names, compression, dimension order, ...

```
cdo cmor,MIP-TABLE
infile
```

Modular post-processing - Implementation

Aggregation | Diagnostics **cdo expr** | Standardisation **cdo cmor**

Aggregation

- Not supported by WebGUI
- Most computation time consuming

→

Diagnostics + Standardisation

- Automatic creation of script fragments by parsing mapping and recipe table information
  - No user generated script with calls to cdo operators

Modular post-processing - Implementation

Aggregation | Diagnostics **cdo expr** | Standardisation **cdo cmor**

Aggregation

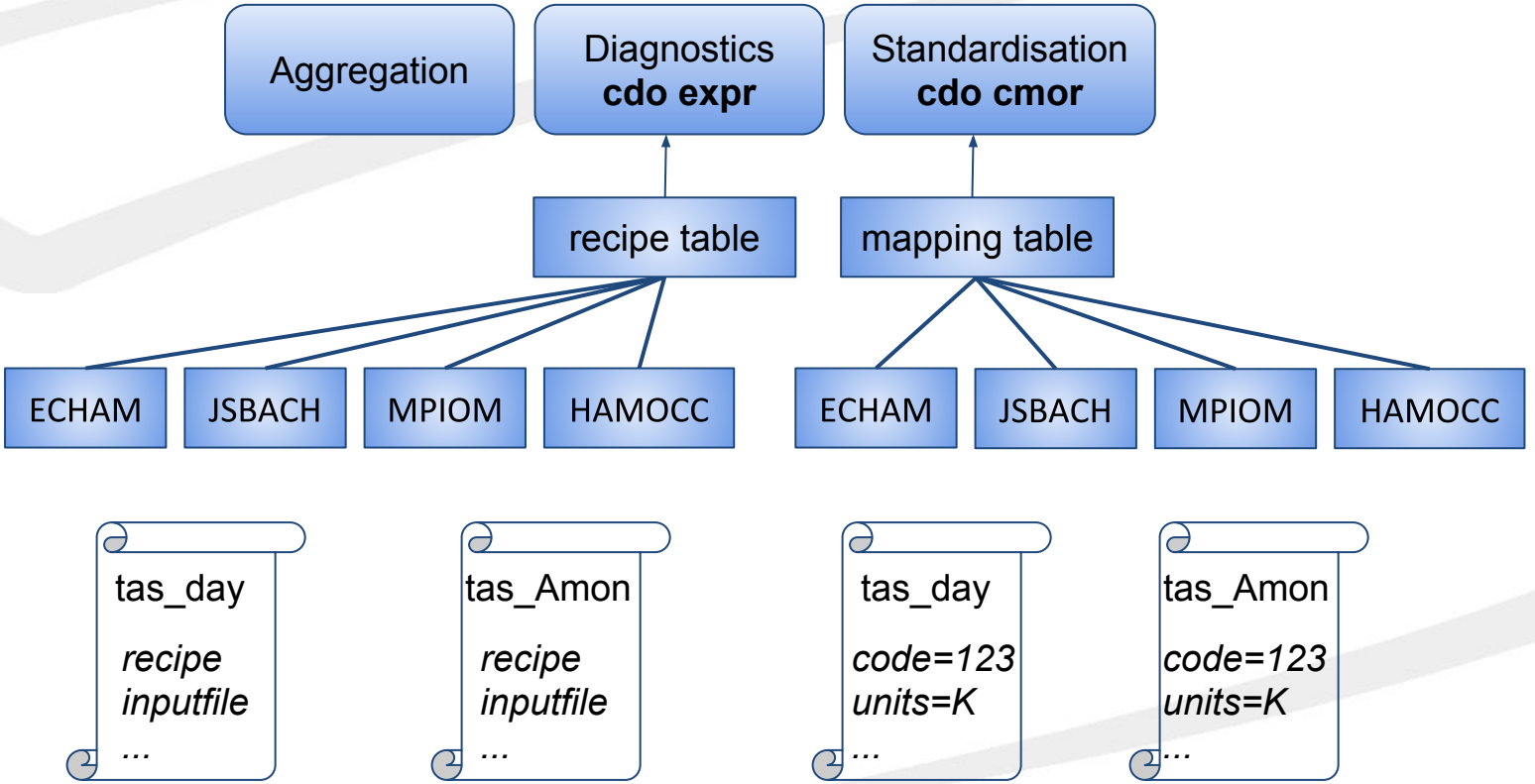
- Not supported by WebGUI
- Most computation time consuming

WebGUI based post-processing **ONLY easily applicable if** raw output file names (namelist) are consistent over experiments!

Diagnostics + Standardisation

- Automatic creation of script fragments by parsing mapping and recipe table information
  - No user generated script with calls to cdo operators

Modular post-processing - Organization



Module

Submodel

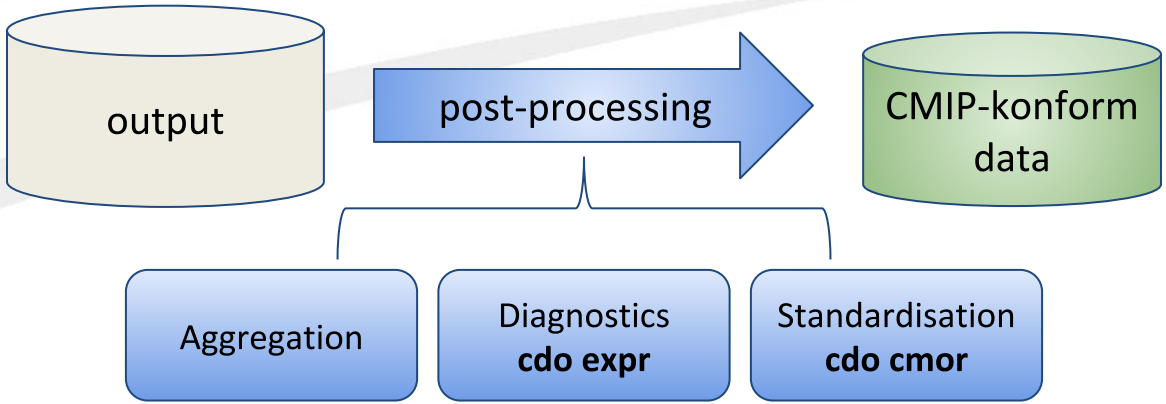
CMOR Variable

Workshop on CMIP6 Post-processing

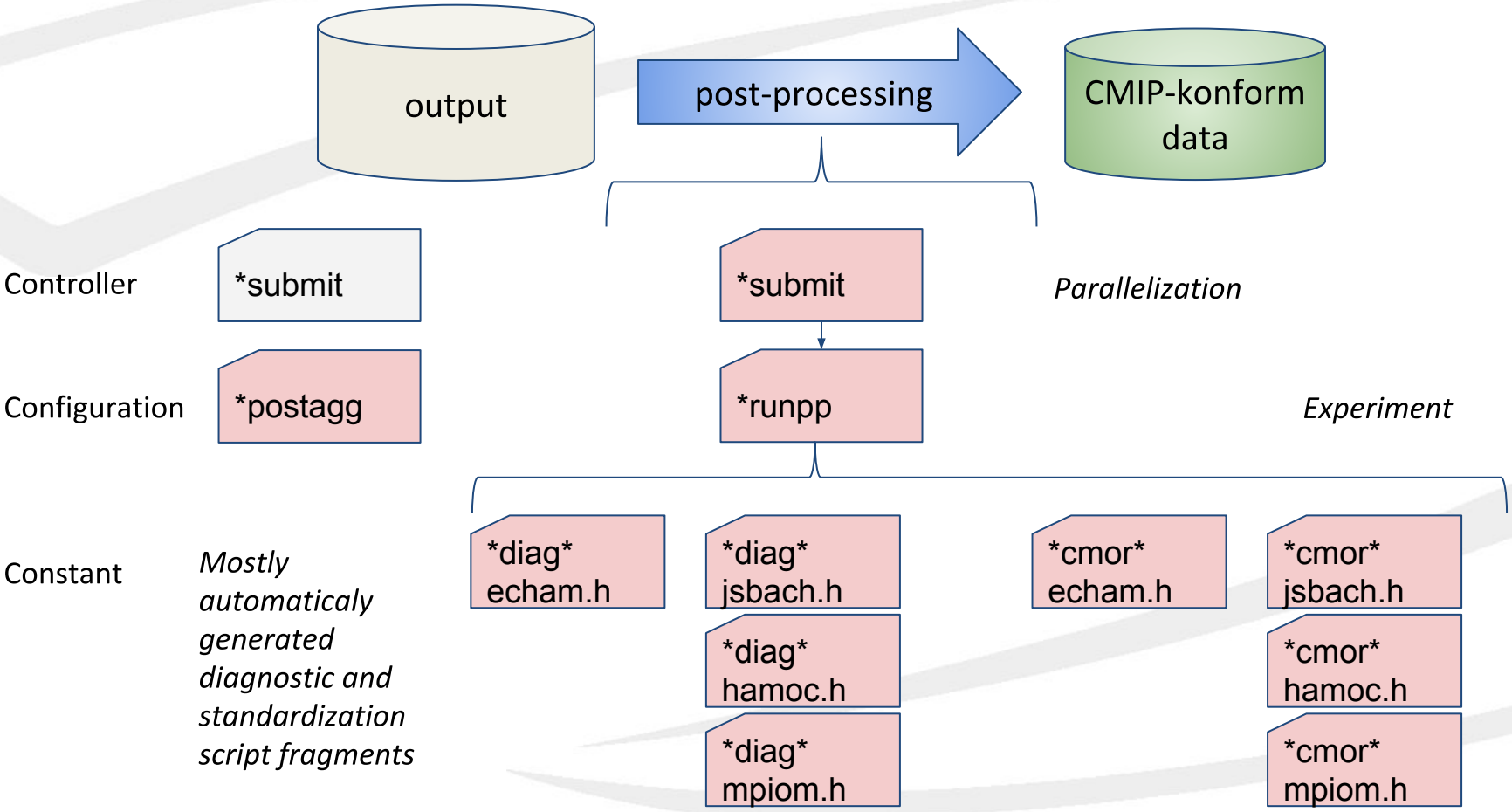
Reenacting the post-processing for historical_r1i1p1f1-LR

Martin Schupfner (schupfner@dkrz.de),
Fabian Wachsmann (wachsmann@dkrz.de),
DKRZ

Reenactment of the Post-processing for historical_r1i1p1f1-LR



Reenactment of the Post-processing for historical_r1i1p1f1-LR



Exercise 1-2 - Preparation

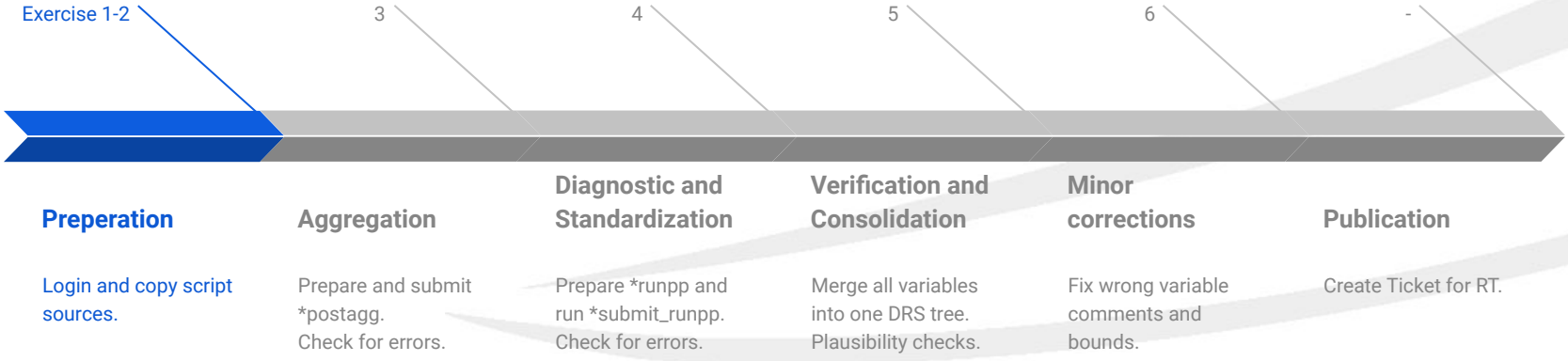
1) Login to mistral with one of the temporary workshop accounts

ssh -X k123456@mistral.dkrz.de

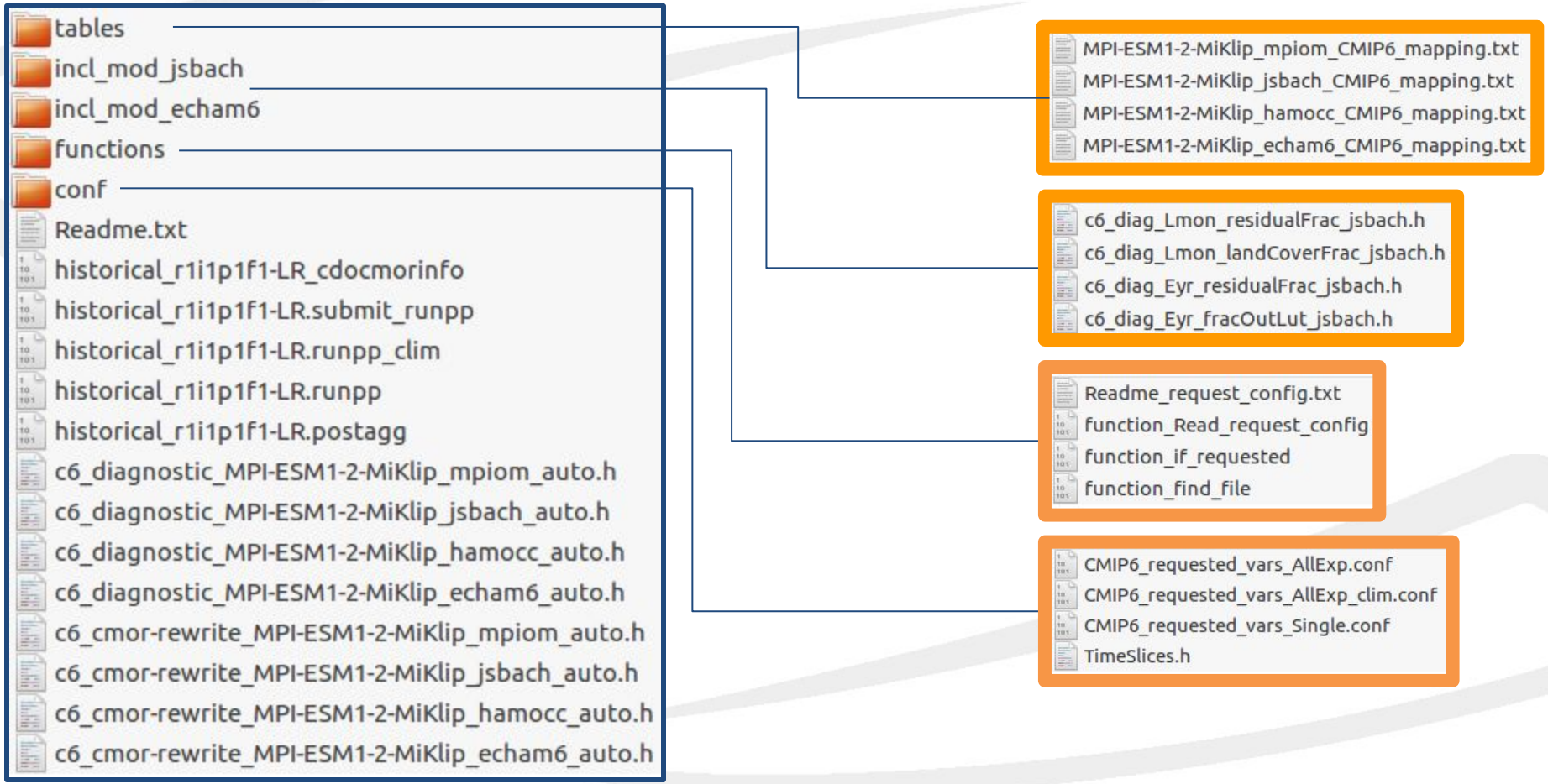
2) Copy the following folder to your work or scratch directory:
/work/bm0021/workshopcmip6pp2019/to_copy/historical_r1i1p1f1-LR

cp -r /work/bm0021/workshopcmip6pp2019/to_copy/historical_r1i1p1f1-LR /scratch/k/k123456

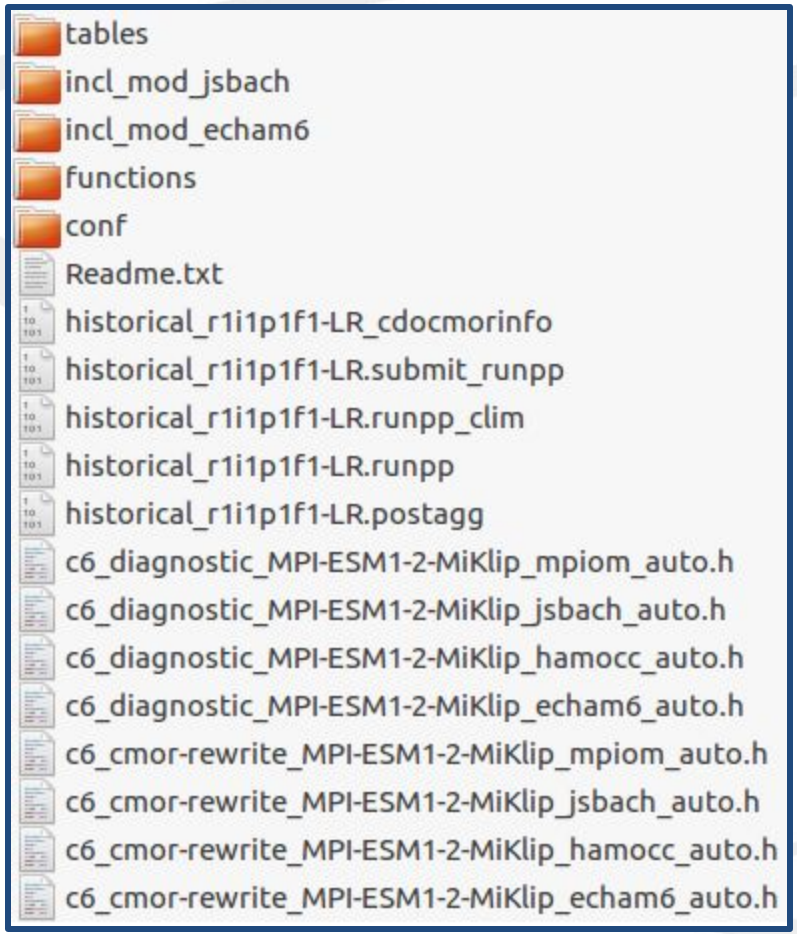
cd /scratch/k/k123456/historical_r1i1p1f1-LR/scripts



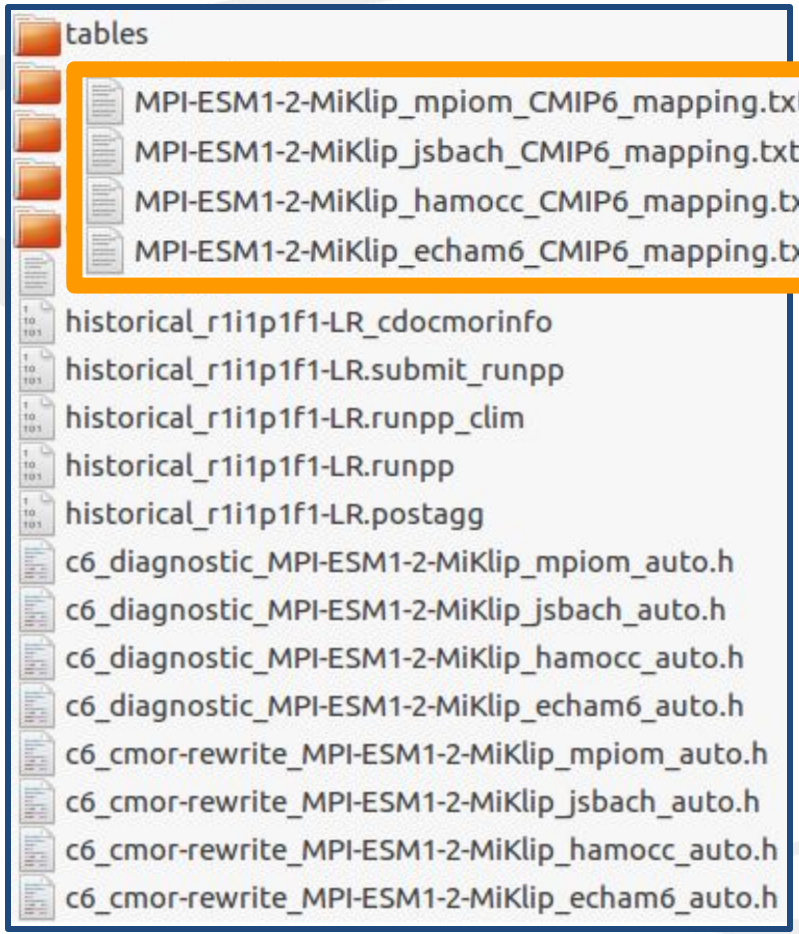
CMIP6 DICAD Post-processing - scripts and configuration



CMIP6 DECK/historical Post-processing - files

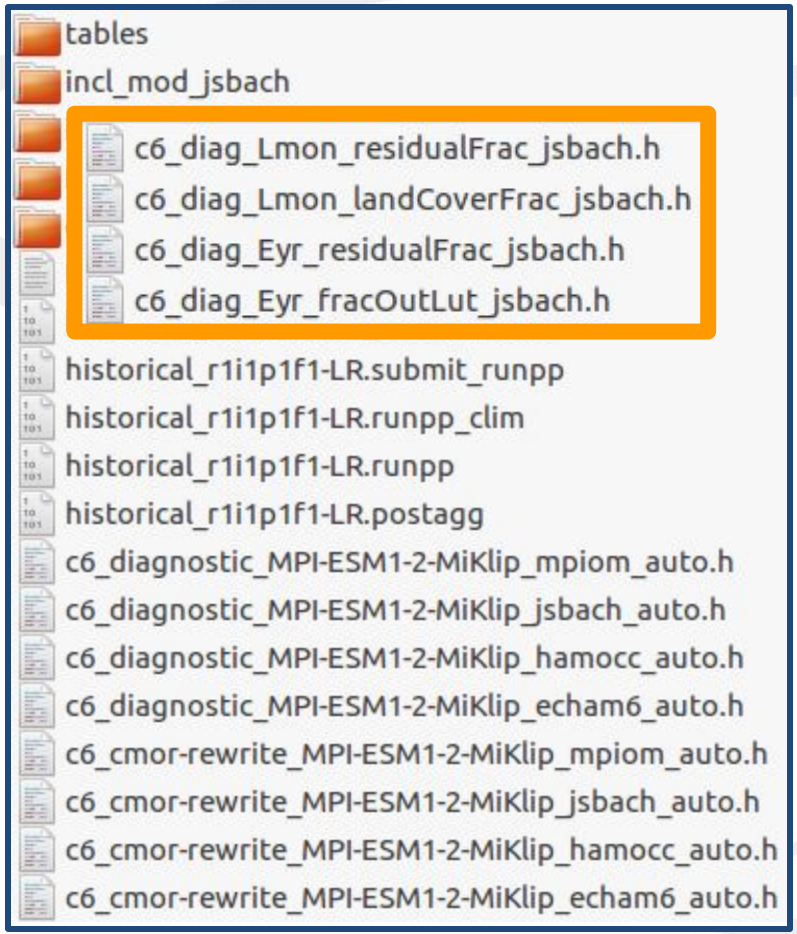


CMIP6 DECK/historical Post-processing - files



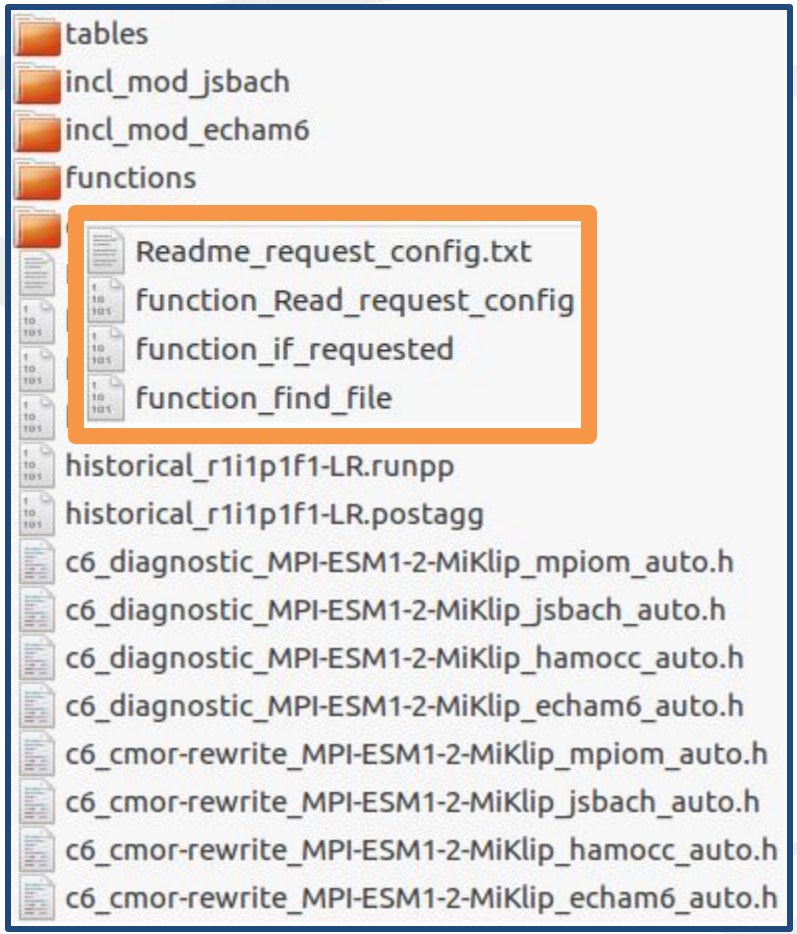
- tables/
  mapping tables in .txt format

CMIP6 DECK/historical Post-processing - files



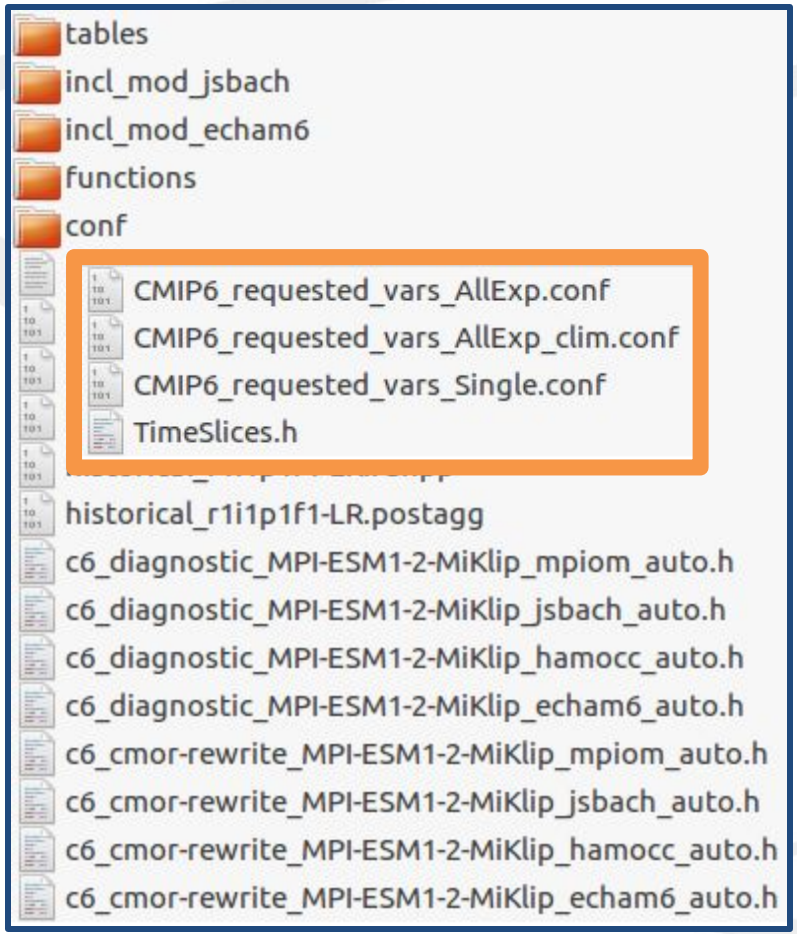
- tables/
  mapping tables in .txt format
- incl_mod_<submodel>/
  custom diagnostic script fragments

CMIP6 DECK/historical Post-processing - files



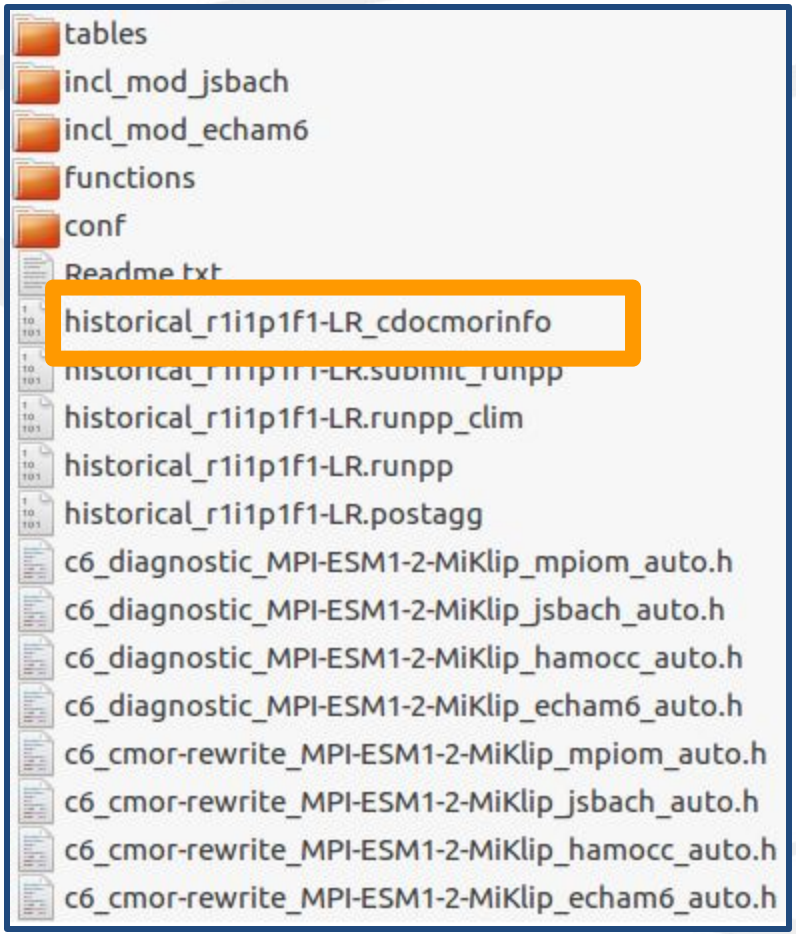
- tables/
  mapping tables in .txt format
- incl_mod_<submodel>/
  custom diagnostic script fragments
- functions/
  korn shell functions

CMIP6 DECK/historical Post-processing - files



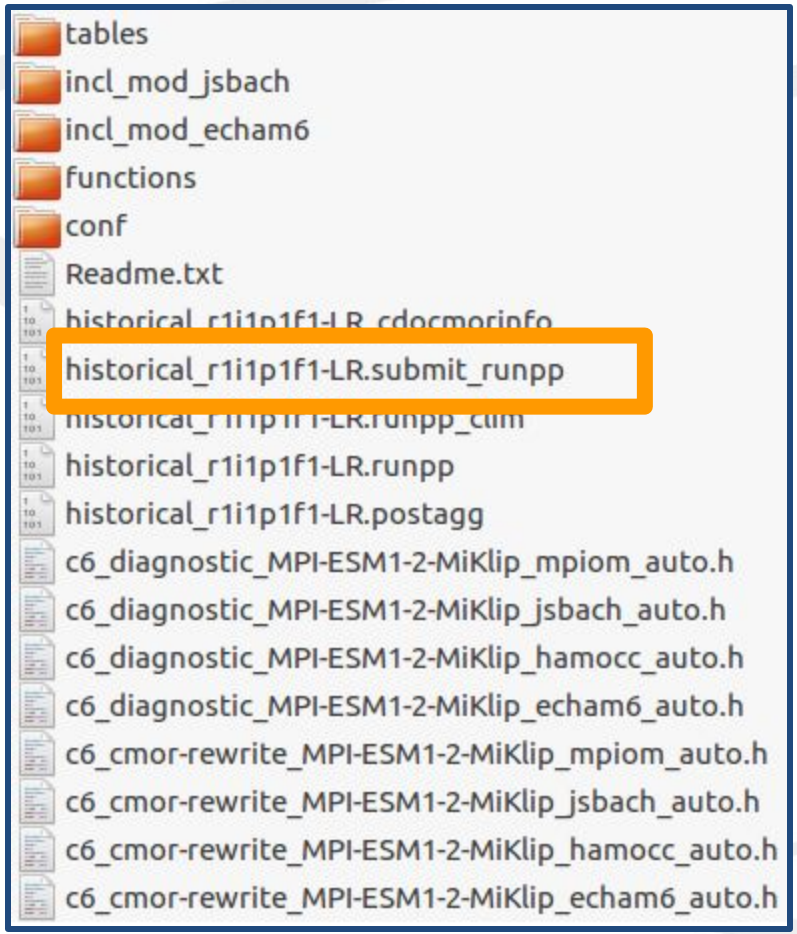
- tables/
  mapping tables in .txt format
- incl_mod_<submodel>/
  custom diagnostic script fragments
- functions/
  korn shell functions
- conf/
  Requested vars configuration for each experiment
  CMIP6 time slices definitions

CMIP6 DECK/historical Post-processing - files



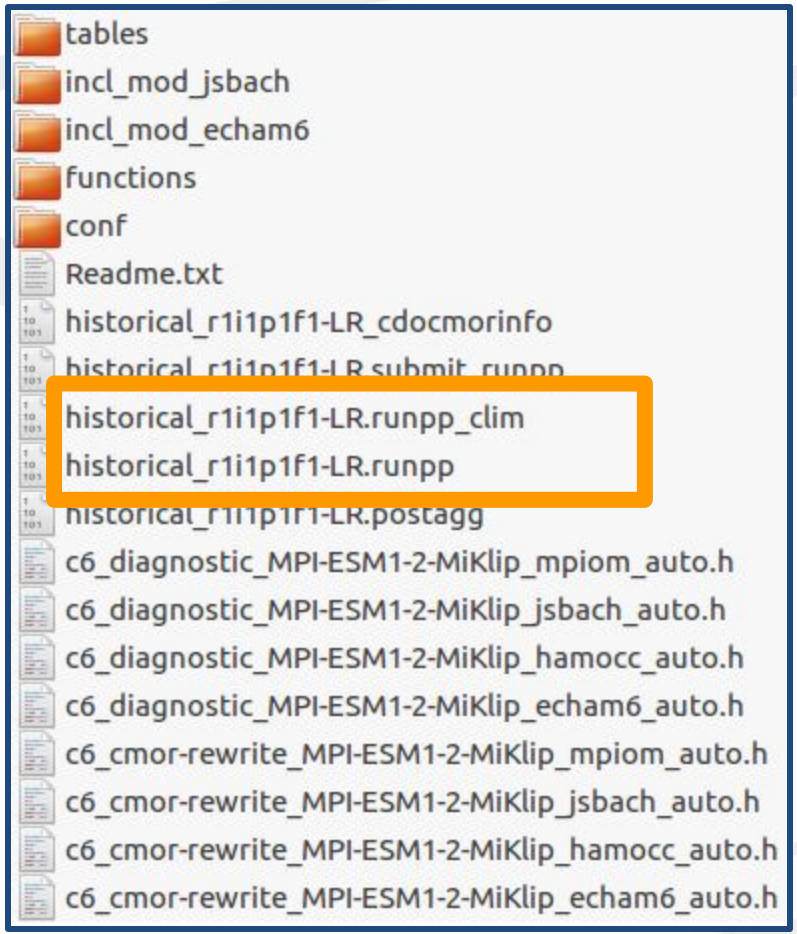
- tables/
  mapping tables in .txt format
- incl_mod_<submodel>/
  custom diagnostic script fragments
- functions/
  korn shell functions
- conf/
  Requested vars configuration for each experiment
  CMIP6 time slices definitions
- cdo cmor info table (to configure necessary global attributes)

CMIP6 DECK/historical Post-processing - files



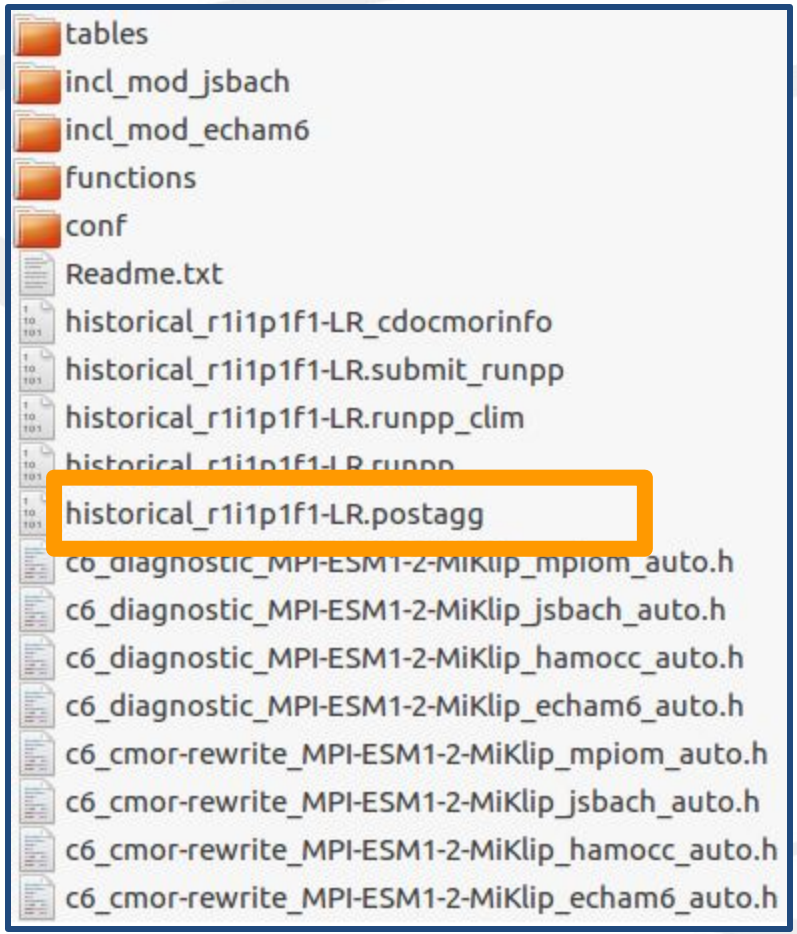
- tables/
  mapping tables in .txt format
- incl_mod_<submodel>/
  custom diagnostic script fragments
- functions/
  korn shell functions
- conf/
  Requested vars configuration for each experiment
  CMIP6 time slices definitions
- cdo cmor info table (to configure necessary global attributes)
- post processing submit script (to submit postprocessing chunks)

CMIP6 DECK/historical Post-processing - files



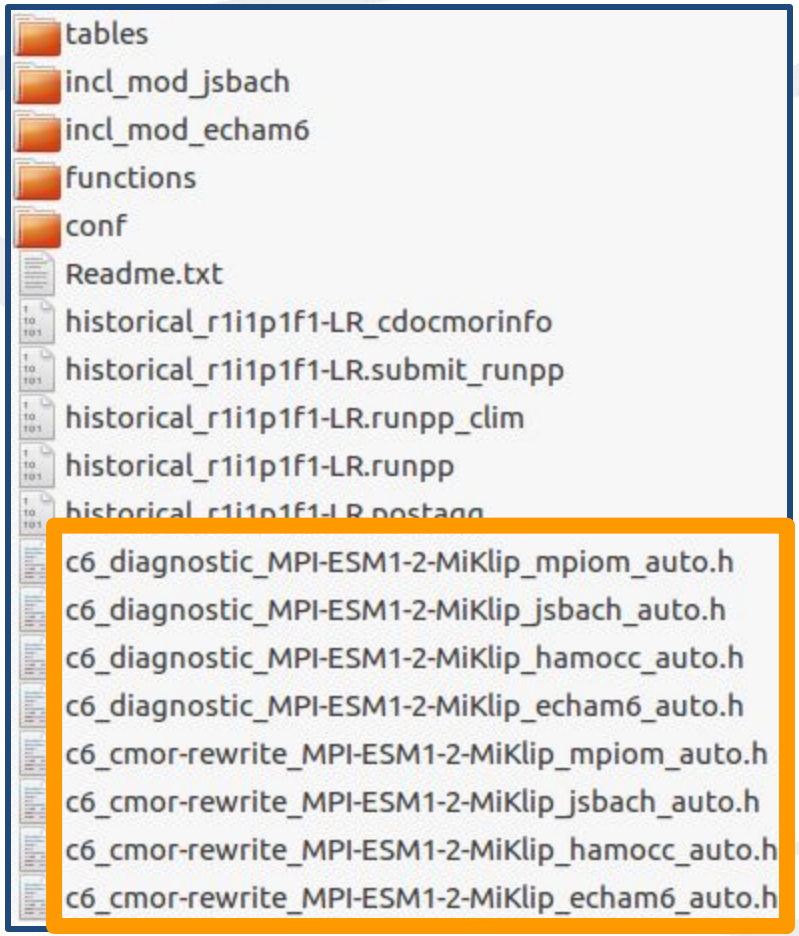
- tables/
  mapping tables in .txt format
- incl_mod_<submodel>/
  custom diagnostic script fragments
- functions/
  korn shell functions
- conf/
  Requested vars configuration for each experiment
  CMIP6 time slices definitions
- cdo cmor info table (to configure necessary global attributes)
- post processing submit script (to submit post processing scripts)
- post processing scripts

CMIP6 DECK/historical Post-processing - files



- tables/
  mapping tables in .txt format
- incl_mod_<submodel>/
  custom diagnostic script fragments
- functions/
  korn shell functions
- conf/
  Requested vars configuration for each experiment
  CMIP6 time slices definitions
- cdo cmor info table (to configure necessary global attributes)
- post processing submit script (to submit post processing scripts)
- post processing scripts
- aggregation script

CMIP6 DECK/historical Post-processing - files

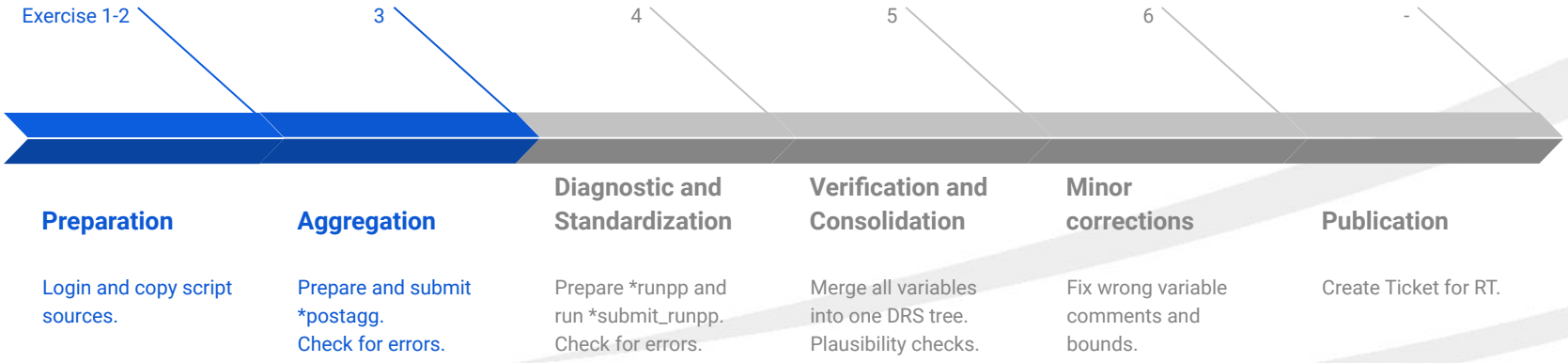


- tables/
  mapping tables in .txt format
- incl_mod_<submodel>/
  custom diagnostic script fragments
- functions/
  korn shell functions
- conf/
  Requested vars configuration for each experiment
  CMIP6 time slices definitions
- cdo cmor info table (to configure necessary global attributes)
- post processing submit script (to submit post processing scripts)
- post processing scripts
- aggregation script
- diagnostic and cmor-rewrite script fragments (called by post processing scripts)

Exercise 3 - Diagnostic and Standardization

At first, adapt the paths in the aggregation script (*.postagg) and submit it for the year 1850. The script performs necessary mean building and vertical interpolations as well as some cdo and cdo afterburner calls.

If the script has completed, see if it was successful for all variables (stdout and stderr for each aggregation block is getting logged in ./errors_agg/).



Exercise 3 - Aggregation

At first, adapt the paths in the aggregation script (*.postagg) and submit it for the year 1850. The script performs necessary mean building and vertical interpolations as well as some cdo and cdo afterburner calls.

1. Open historical_r1i1p1f1-LR.postagg in an editor (gedit, vim, emacs, …)
2. Adjust all paths necessary (as well as add the QOS if possible).
3. Submit the script via:
   sbatch historical_r1i1p1f1-LR.postagg 18500101
4. Monitor the slurm queue using for example:
   squeue -u k123456

If the script has completed, see if it was successful for all variables (stdout and stderr for each aggregation block is getting logged in ./errors_agg/).

1. Grep for ERROR:
   grep -R ERROR ./errors_agg/

Script fragments

Diagnostic
c6_diagnostic.h

- **cdo merge** (of multiple inputfiles)
- **cdo expr**

```
#-- Diagnostic for echam6 (ESM: AWI-CM-1-0-HR) variable rlds / table 3hr
{ (if_requested $member $atmmod 3hr rlds $chunk && {
  find_file -e -p $period "$sdir" "TST_????01.01_echam.grb" ifile1
  find_file -e            "$sdir" "rlds_3hr_${period}*" ifile2
  $cdo -f nc -0 \
    expr,'rlds=var177-var205;' \
    -merge -selcode,177 $ifile1 -selcode,205 $ifile2 \
    in_cmor/3hr_rlds_$period.nc || echo ERROR
}; )&; }>$err.rlds.3hr 2>&1
```

CMOR rewrite
c6_cmor-rewrite.h

- **cdo cmor** call

```
#-- CMOR-rewrite for echam6 (ESM: AWI-CM-1-0-HR) 3hr
cn='rlds hfls'
for var in $cn; do
  { (if_requested $member $atmmod 3hr $var $chunk || continue
  ifile=in_cmor/3hr_${var}_$period.nc
  $cdo cmor,3hr,mt=$mt,dr=$dr,cn=$var $ifile || echo ERROR
  )&; }>>$err.$var.3hr 2>&1
done
```

The script fragment does ...

- … test if variable is requested (data request, timeslice, user specifications)
- … find input files & call cdo

Script fragments

Diagnostic
c6_diagnostic.h

- **cdo merge** (of multiple inputfiles)
- **cdo expr**

```
#   Diagnostic for echam6 (ESM: AWI-CM-1-0-HR) variable rlds / table 3hr
{ (if_requested $member $atmmod 3hr rlds $chunk && {
  find_file -e -p $period "$sdir" "ISI_??????01.01_echam.grb" ifile1
  find_file -e            "$sdir" "rlds_3hr_${period}*" ifile2
  $cdo -f nc -0 \
    expr,'rlds=var177-var205;' \
    -merge -selcode,177 $ifile1 -selcode,205 $ifile2 \
    in_cmor/3hr_rlds_$period.nc || echo ERROR
}; )&; }>$err.rlds.3hr 2>&1
```

CMOR rewrite
c6_cmor-rewrite.h

- **cdo cmor** call

```
#-- CMOR-rewrite for echam6 (ESM: AWI-CM-1-0-HR) 3hr
cn='rlds hfls'
for var in $cn; do
  { (if_requested $member $atmmod 3hr $var $chunk || continue
  ifile=in_cmor/3hr_${var}_$period.nc
  $cdo cmor,3hr,mt=$mt,dr=$dr,cn=$var $ifile || echo ERROR
  )&; }>>$err.$var.3hr 2>&1
done
```

The script fragment does ...

- … test if variable is requested (data request, timeslice, user specifications)
- … find input files & call cdo

Script fragments

Diagnostic
c6_diagnostic.h

- **cdo merge** (of multiple inputfiles)
- **cdo expr**

```
#-- Diagnostic for echam6 (ESM: AWI-CM-1-0-HR) variable rlds / table 3hr
{ (if_requested $member $atmmod 3hr rlds $chunk && {
  find_file -e -p $period "$sdir" "TST_????01.01_echam.grb" ifile1
  find_file -e            "$sdir" "rlds_3hr_${period}*" ifile2
  $cdo -f nc -O \
    expr,'rlds=var177-var205;' \
    -merge -selcode,177 $ifile1 -selcode,205 $ifile2 \
    in_cmor/3hr_rlds_$period.nc || echo ERROR
}; )&; }>$err.rlds.3hr 2>&1
```

CMOR rewrite
c6_cmor-rewrite.h

- **cdo cmor** call

```
#-- CMOR-rewrite for echam6 (ESM: AWI-CM-1-0-HR) 3hr
cn='rlds hfls'
for var in $cn; do
  { (if_requested $member $atmmod 3hr $var $chunk || continue
  ifile=in_cmor/3hr_${var}_$period.nc
  $cdo cmor,3hr,mt=$mt,dr=$dr,cn=$var $ifile || echo ERROR
  )&; }>>$err.$var.3hr 2>&1
done
```

The script fragment does ...

- … test if variable is requested (data request, timeslice, user specifications)
- … find input files & call cdo

Script fragments

Diagnostic
c6_diagnostic.h

- **cdo merge** (of multiple inputfiles)
- **cdo expr**

```
#-- Diagnostic for echam6 (ESM: AWI-CM-1-0-HR) variable rlds / table 3hr
{ (if_requested $member $atmmod 3hr rlds $chunk && {
  find_file -e -p $period "$sdir" "TST_????01.01_echam.grb" ifile1
  find_file -e -p $period "$sdir" "rlds_3hr_*.nc" ifile2
  $cdo -f nc -0 \
    expr,'rlds=var177-var205;' \
    -merge -selcode,177 $ifile1 -selcode,205 $ifile2 \
    in_cmor/3hr_rlds_$period.nc || echo ERROR
} )&; }>>$err.rlds.3hr 2>&1
```

CMOR rewrite
c6_cmor-rewrite.h

- **cdo cmor** call

```
#-- CMOR-rewrite for echam6 (ESM: AWI-CM-1-0-HR) 3hr
cn='rlds hfls'
for var in $cn; do
  { (if_requested $member $atmmod 3hr $var $chunk || continue
  ifile=in_cmor/3hr_${var}_$period.nc
  $cdo cmor,3hr,mt=$mt,dr=$dr,cn=$var $ifile || echo ERROR
  )&; }>>$err.$var.3hr 2>&1
done
```

The script fragment does ...

- … test if variable is requested (data request, timeslice, user specifications)
- … find input files & call cdo

Post processing script - *.runpp

- Define certain variables (experiment, path, …)

```
#Models
atmmod=echam6
bgcmod=hamocc
ocemod=mpiom
srfmod=jsbach
esmod=MPI-ESM1-2

#Experiment etc
experiment=abrupt-4xCO2
member=r1i1p1f1
mip=CMIP6
```

- Initialize if_requested function and read data request

```
. $fpath/function_if_requested
. $fpath/function_Read_request_config
. $fpath/function_find_file

#Load predefined timeslices (depends on $iniyear/$finyear)
. $cpath/TimeSlices.h


#Initialize DataRequest/User Configuration for the if_requested function
# This will read all SettingsContainers, TimeSlices and the configuration file
Read_request_config -s $experiment ./conf/${mip}_requested_vars_$experiment.conf || exit 1
```

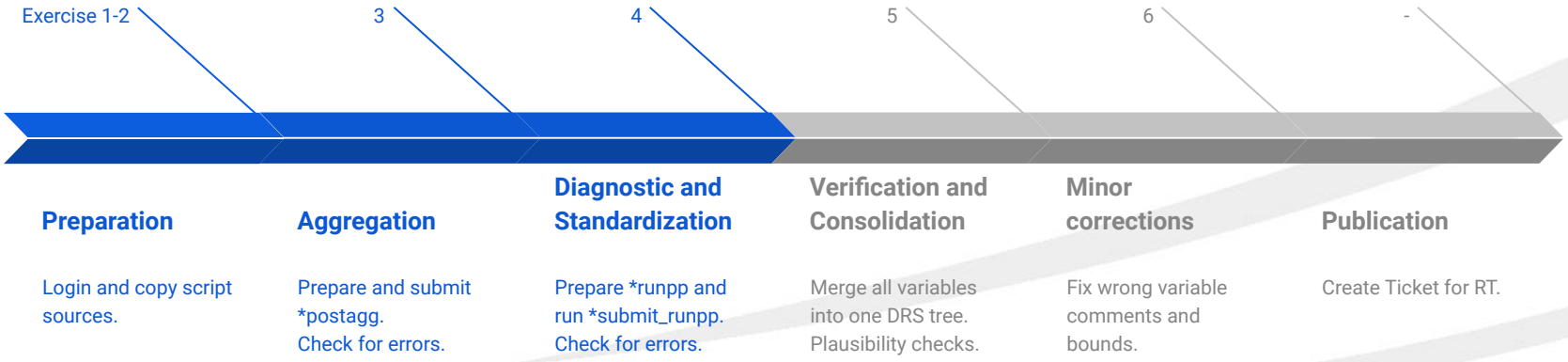
- Run diagnostic and cmor-rewrite script fragments in a time loop

```
#Loop over file output periods and load CMOR-Rewrite script fragment
for period in `seq ${iniyear} ${finyear}`; do
    #Define the chunk that is tested by if_requested (YYYYMMDDHH-YYYYMMDDHH)
    chunk=${period}010100-${period}123124

    for submodel in $atmmod $ocemod $srfmod $bgcmod; do

        #Location of Mapping table:
        mt="./tables/${esmod}_${submodel}_${mip}_mapping.txt"

        #Load and run the CMOR-Rewrite ScriptFragment
        echo "Performing cdo CMOR operations for $submodel $chunk..."
        . ./c6_cmor-rewrite_${esmod}_${submodel}_auto.h
        wait
    done
done
```

Exercise 4 - Diagnostic and Standardization

Now adapt the paths in the post processing script (*.runpp) as well as the paths in the submit post processing script (*.submit_runpp). Then submit the post processing script for the year 1850 by running the *.submit_runpp script.

If the script has completed, see if it was successful for all variables (stdout and stderr for each aggregation block is getting logged in ./errors/).



Exercise 4 - Diagnostic and Standardization

Now adapt the paths in the post processing script (*.runpp) as well as the paths in the submit post processing script (*.submit_runpp). Then submit the post processing script for the year 1850 by running the *.submit_runpp script.

1. Open historical_r1i1p1f1-LR.runpp and historical_r1i1p1f1-LR.submit in an editor (gedit, vim, emacs, …)
2. Adjust all paths necessary (as well as add the QOS if possible).
3. Submit the post processing script via:
   bash historical_r1i1p1f1-LR.submit_runpp
4. Monitor the slurm queue using for example:
   squeue -u k123456

If the script has completed, see if it was successful for all variables (stdout and stderr for each aggregation block is getting logged in ./errors/).

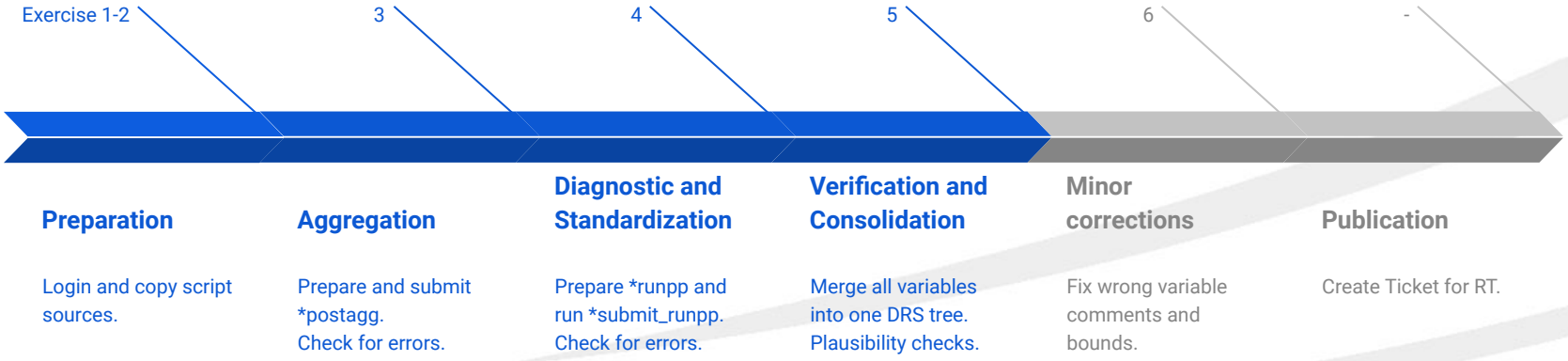
1. Grep for ERROR:
   grep -R ERROR ./errors/

Naming convention of the log files: err<YEAR>.<CMORvar>.<MIPtable>

Exercise 5 - Verification and Consolidation

Merge all cmorized variables into one DRS tree.

You should perform a plausibility check on the cmorized files:
Did all desired variables get produced?
Do the fields look as they are supposed to look?
Do the values and units fit together?



Exercise 5 - Verification and Consolidation

Merge all cmorized variables into one DRS tree.

All variables have been written to the directory:

../archive/<submodel>/<MIPtable>_<CMORvar>/CMIP6/<DRS>

Each variable has its own DRS (CMIP6 conformal directory structure) tree to not interfere during the post processing and allow for an easier debugging. However, for publishing, all need to be in the same DRS tree. To consolidate/merge all "cmorized" variables switch to the archive directory ($dr in the post processing scripts) and merge all files into the same DRS tree:

```
cd ../archive
cp -rl */*_*/CMIP6 .
```

This creates hard links of all the created files in only one DRS tree.
After completion, *../archive/<submodel>* directories can be deleted.

```
rm -r echam6 jsbach hamocc mpiom
```

The cmorized data for all submodels is now stored in ../archive/CMIP6

Exercise 6

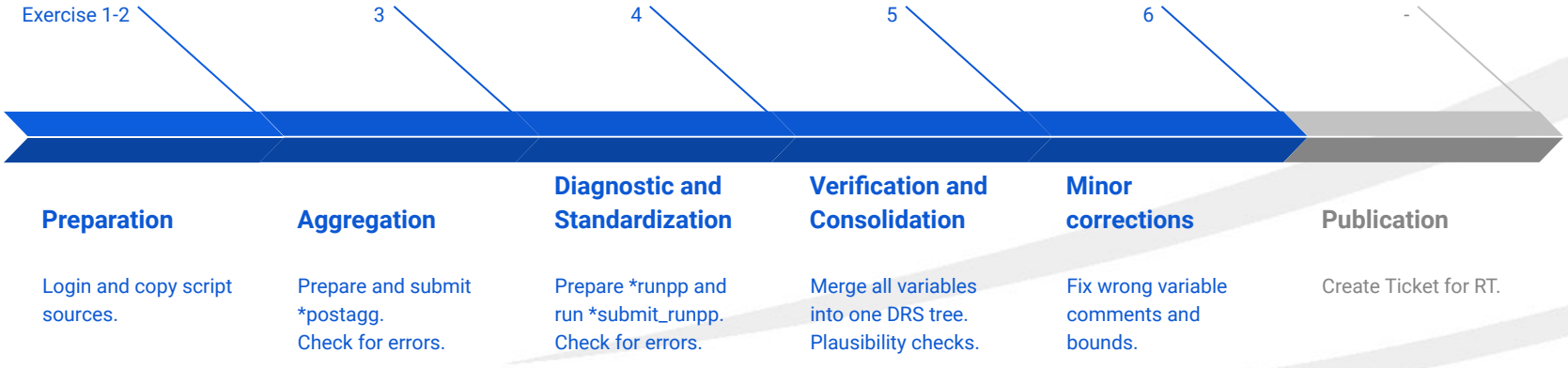
Run the add_bounds.sh and overwrite_comments.sh scripts after adapting the path to fix the bounds and overwrite comments automatically set by CMOR that do not apply for MPI-ESM1-2.

Open both files in an editor and adapt the paths. Finally run them:

module load nco

bash add_bounds.sh

bash overwrite_comments.sh

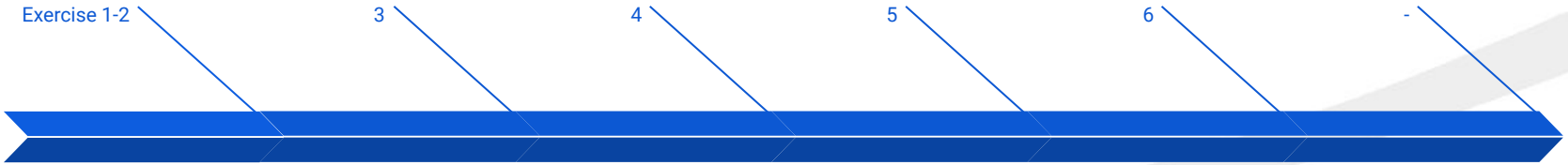


How to request ESGF publishing

Finally, request a Quality Assurance (QA) check and subsequent publishing at the DKRZ ESGF-node. You can open the request by sending an E-Mail including all necessary information to:
esgf-publication@dm-rt.dkrz.de

Please provide the path to the cmorized files as well as the names of the conducted experiment, realization (= member id) and your climate model.

The QA will find any errors in the data standard, but performs also minor consistency and plausibility checks. If the QA is successful, the data will be published.



(Exercise 7)

Adapt the paths in historical_r1i1p1f1-LR.runpp_clim, create a custom climatology/time slice for Oclim variables for the years 1850-1854 in ./conf/CMIP6_requested_vars_AllExp_clim.conf and run historical_r1i1p1f1-LR.runpp_clim for this time interval.

Open both files in an editor and make the changes. Finally run:

sbatch historical_r1i1p1f1-LR.runpp_clim

```
################################################
EXP=historical
################################################

DREQSETTINGS
Emon        : Emon       = slice: piControl030,piControl050
Emon        : thetaot300 = False

USERSETTINGS
Field       : Key        = True/False
Field       : Key        = slice:YYYYMMDDHH-YYYYMMDDHH
```

Requested variables configuration file:

- Experiment must be referenced in the post processing script
- CMIP6 time slices listed in ./conf/TimeSlices.h
- User settings are given priority
- Field/Key can be (from high to low priority):
  - CMOR variable name (eg. “tas”)
  - MIP table name (eg. “Amon”)
  - submodel (eg. “echam6”)
  - member id (eg. “r2i1p1f1”)
  - “TOTAL”

Questions?

To keep in mind:

It really is tedious to adapt the entire scripting to each single experiment, rather than just change a few variables / paths in the scripts. So please:

- **Any custom namelist changes for your experiment (in comparison to the modeling groups' default namelist for CMIP6) will increase the work you need to invest into the post processing, as the mapping, aggregation and post processing scripts will need to be adapted.**
- **If namelist changes are needed that add further model output, add them to the official CMIP6 namelist of your modeling group (if possible), so others can benefit from the additional output and no inconsistencies will be introduced.**
- **Your colleagues might also be interested in your experiments' outcome and certain variables, especially as some MIPs require output from the experiments of other MIPs. This should be discussed before starting the experiment.**

Agenda

10th October 2019

09:15	**Introduction to the CMIP6 Data Request** DreqPy Software, c6dreq WebGUI
09:30	**CMIP6 production workflow** Usage of the Dreq Modular Post-Processing: Aggregation, Diagnostic, Standardizing *Recapitulation*
09:45	**cdo cmor: Introduction, application, configuration** Installation, first steps, data streams CMIP6 data standard, MIP-tables, variable mapping
11:15	*Coffee break*
11:30	**c6dreq WebGUI: Variable Mapping** Variable mapping: DReq updates, logs, diagnostik recipes
12:30	*Lunch break (nothing centrally organized, but mensa is nearby)*
13:15	**cdo cmor: Advanced** Global attributes and info tables ("cdocmorinfo"). coordinate configuration. ESGF-specialities: `tracking_prefix`, `version_date`. How to keep input file attributes ("kaa"). No ESGF publishing required? Define your own data standard. Append mode.
14:30	**c6dreq WebGUI: Script fragments and post processing script** Script fragments composition. Configuration of requested variables. Configuration of the post processing script. Output analysis. *Part I*
15:00	*Coffee break*
15:15	**c6dreq WebGUI: Script fragments** *Part II*
16:00	Official end: Q&A or personal support if desired

Workshop on CMIP6 Post-processing

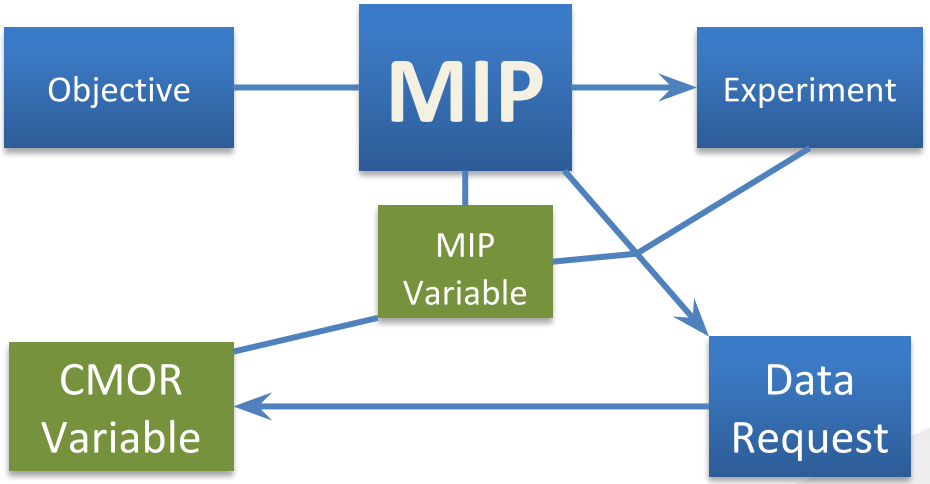
Introduction to the CMIP6 Data Request (Dreq)

Martin Schupfner (schupfner@dkrz.de),
Fabian Wachsmann (wachsmann@dkrz.de),
DKRZ

Oct 10th 2019

CMIP6 Data Request - What is a “CMOR variable”?

- MIPs founded to achieve WCRP defined scientific objectives
- MIPs define Experiments, Variables and set up a data request
- CMOR-Variables are the different versions (frequency, shape, …) of a MIP-Variable



Example:

MIP-Variable: Air Temperature

CMOR Variables:

MIP tables:

(1) **ta** - Air Temperature (zonal mean on 39 pressure levels, monthly mean) - **Emon**

(2) **ta** - Air Temperature (global field on model levels, monthly mean) - **CFmon**

(3) **ta** -Air Temperature (global field on 19 pressure levels, monthly mean) - **Amon**

CMIP6 Data Request Publication

The Data Request is published as:

- SVN repository:
  - 2 xml-files
  - DreqPy (Python API)
- All-In-One Excel Sheet (created using DreqPy)

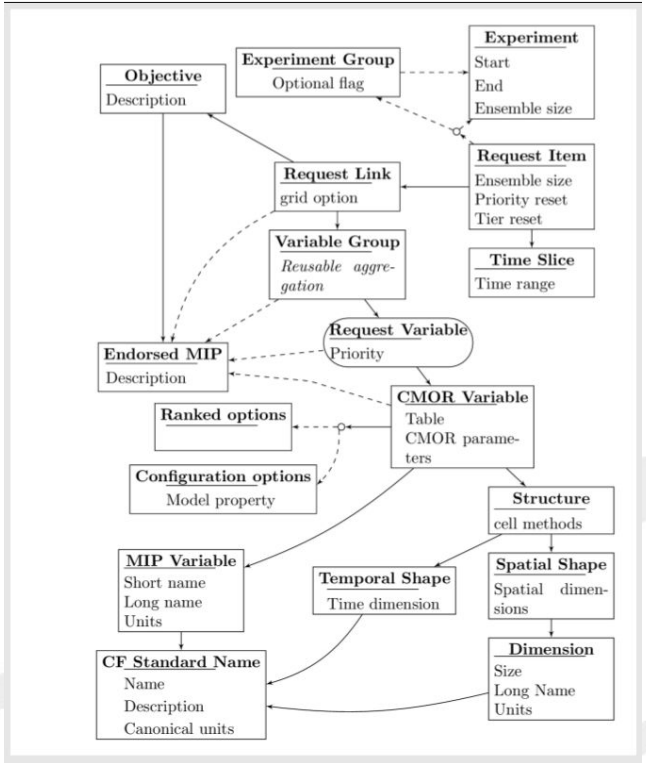
Coordinated by WIP
(WGCM Infrastructure Panel)

https://earthsystemcog.org/projects/wip/CMIP6DataRequest

DreqPy API

by Martin Juckes, BADC

- Interface for the CMIP6 data request written in Python
- Build customized data request (depending on MIP, experiment, variable priority, experiment tier)
- Interactive browsing of the data request possible via a collection of classes and functions
- Calculate data volume estimates
- Output as .csv file and excel sheet



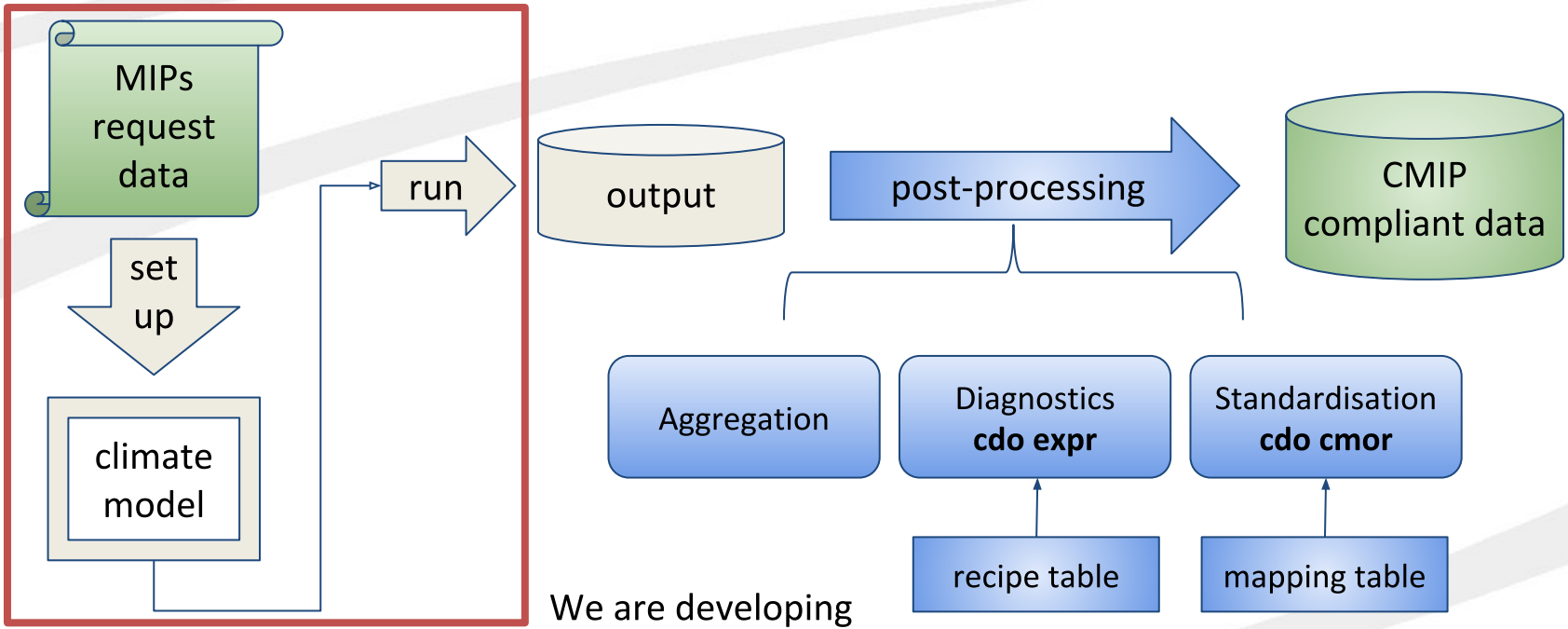
Martin Juckes, Data Request Python API (2015-)

Workshop on CMIP6 Post-processing

From WebGUI to runpp script

Martin Schupfner (schupfner@dkrz.de),
Fabian Wachsmann (wachsmann@dkrz.de),
DKRZ

CMIP6 data production workflow

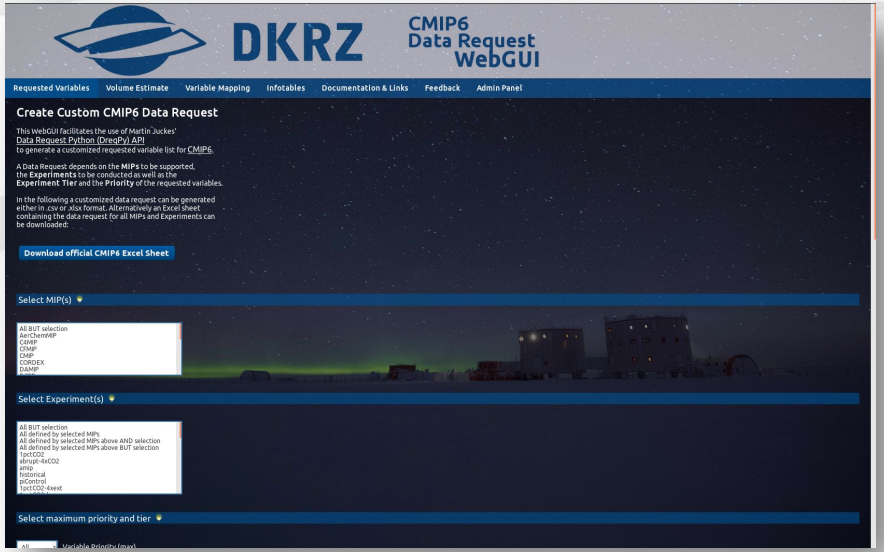


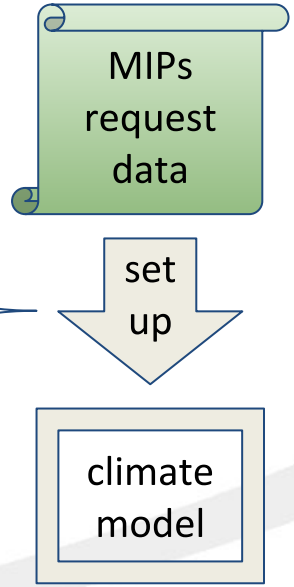
We are developing

- the **cdo cmor** operator
- the **c6dreq-WebGUI** to
  - create recipe/mapping tables
  - create a customized DReq
  - generate script fragments for diagnostic & standardisation

Further steps are not displayed (Monitoring, QA, Citation, Publication, Long-term archiving, …)

Model output configuration





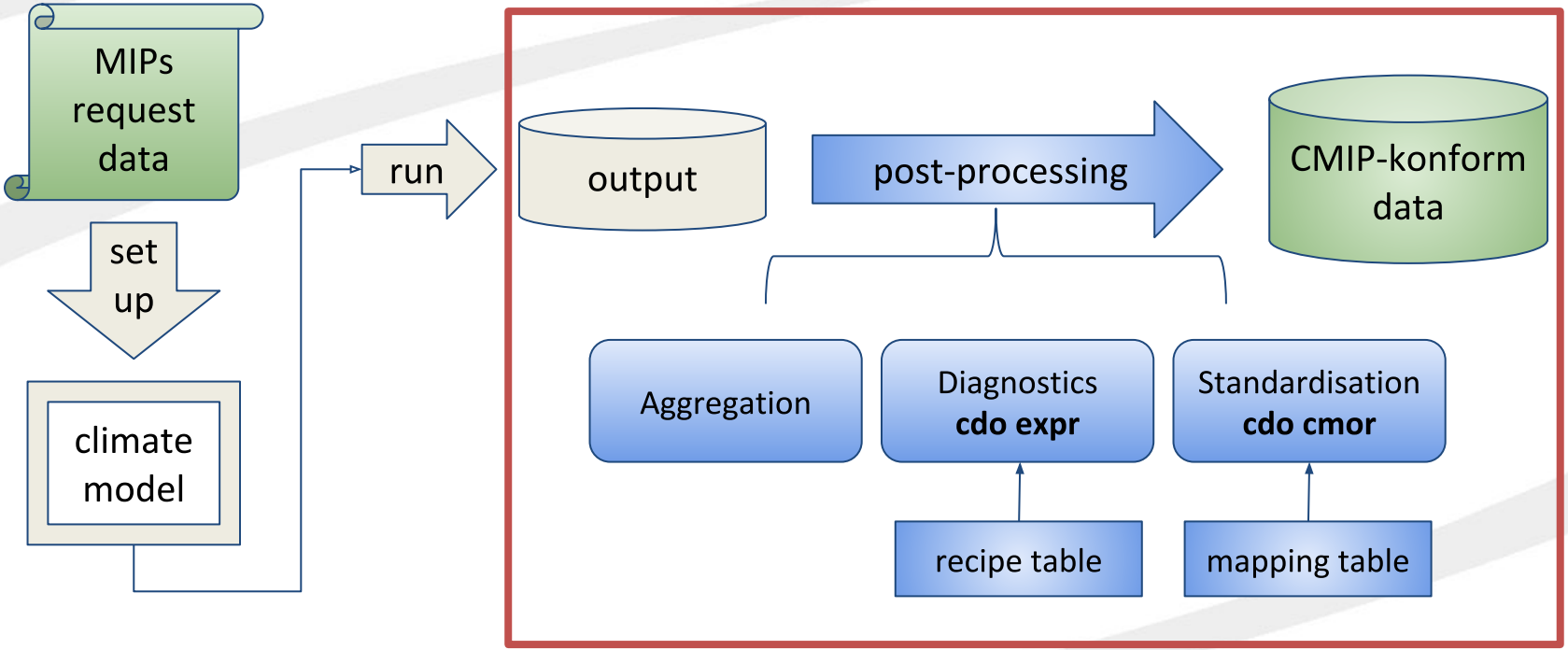
Web GUI to use the basic functions of the Data Request Python API:

- Data request as csv list
- Data request as excel document
- Volume estimate

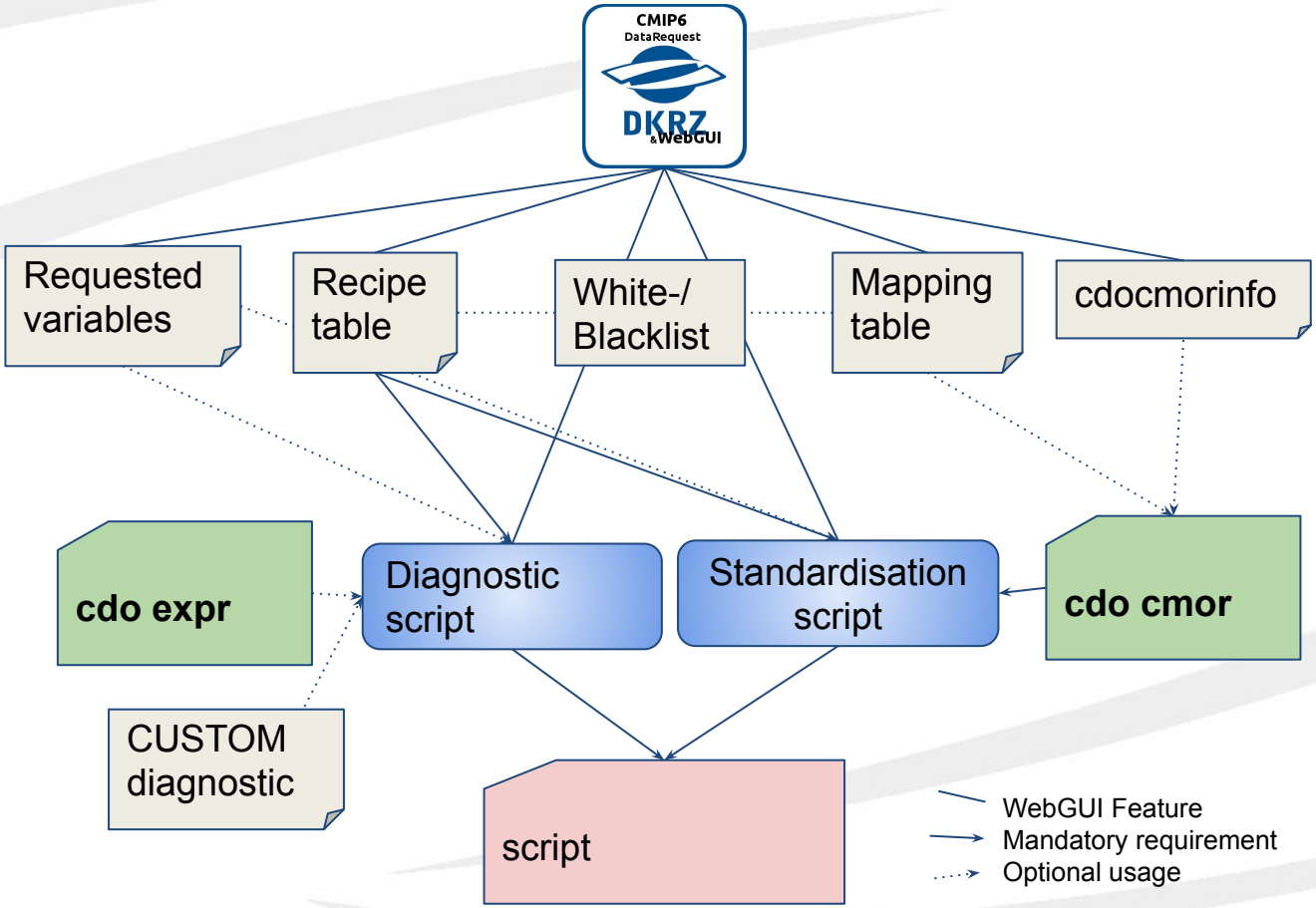
URL: https://c6dreq.dkrz.de

HowTo: https://c6dreq.dkrz.de/info/howto2.php

Modular post-processing

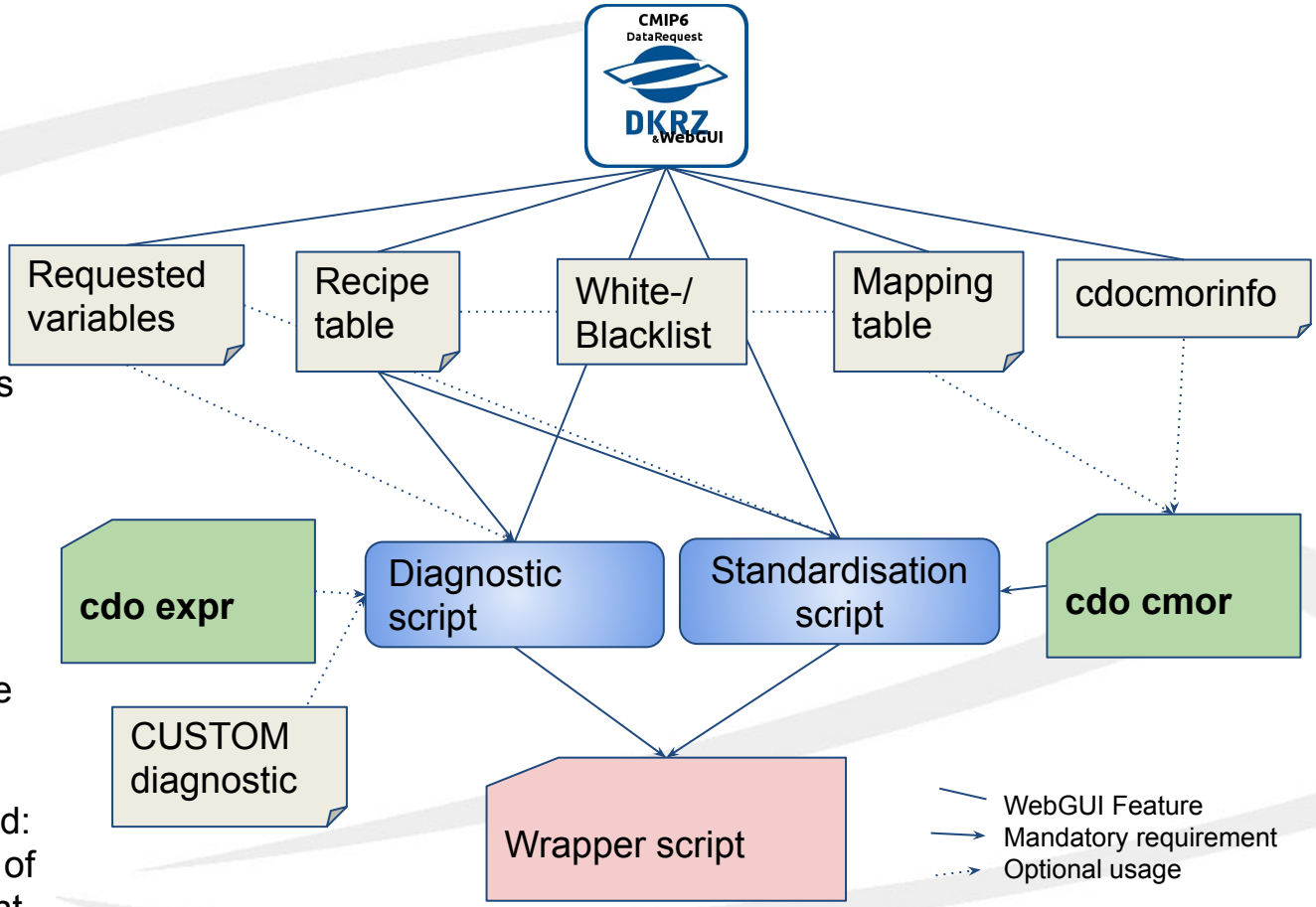


Modular post-processing - suggested practise



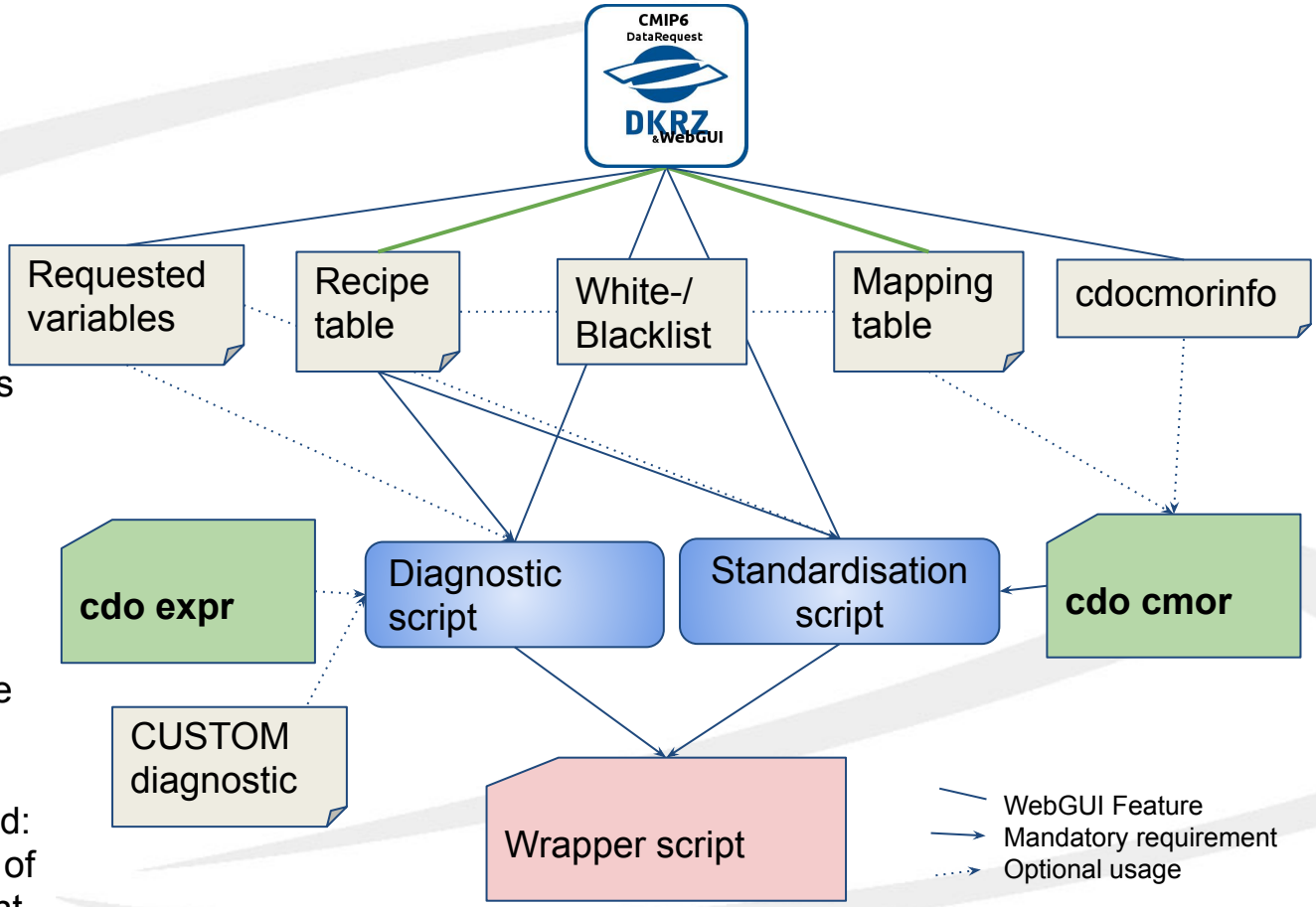
From WebGUI to runpp script

1. Fill in the tables for all CMOR variables which can be delivered
2. Generate script fragments
3. Generate list of global attributes for cdo cmor
4. Include script fragments in a post-processing script
5. There are methods to exclude variables:
   a. Requested variables configuration (one per experiment)
   b. White-/Blacklist
   c. Not recommended: remove parts out of the script fragment



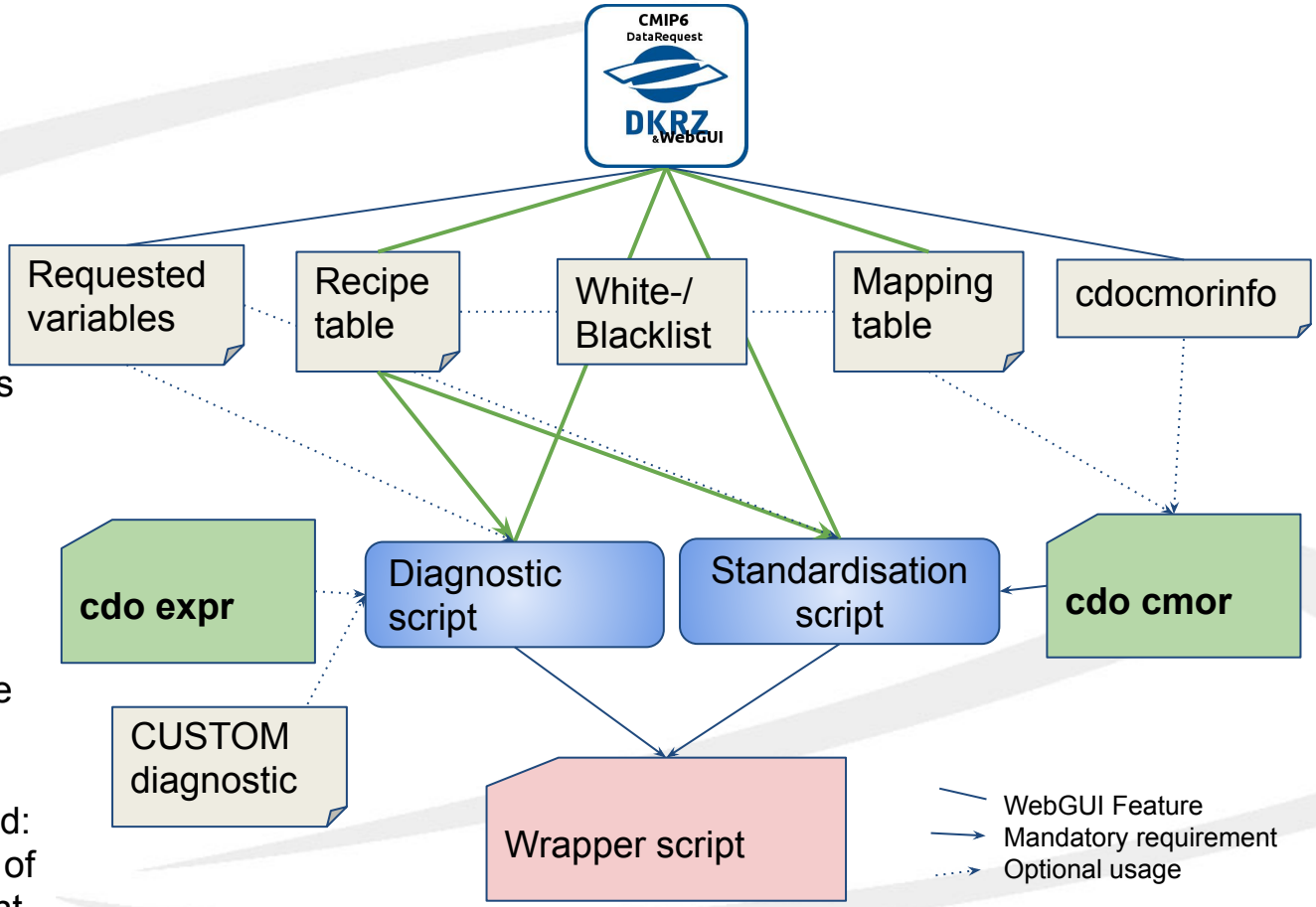
From WebGUI to runpp script

1. Fill in the tables for all CMOR variables which can be delivered
2. Generate script fragments
3. Generate list of global attributes for cdo cmor
4. Include script fragments in a post-processing script
5. There are methods to exclude variables:
   a. Requested variables configuration (one per experiment)
   b. White-/Blacklist
   c. Not recommended: remove parts out of the script fragment



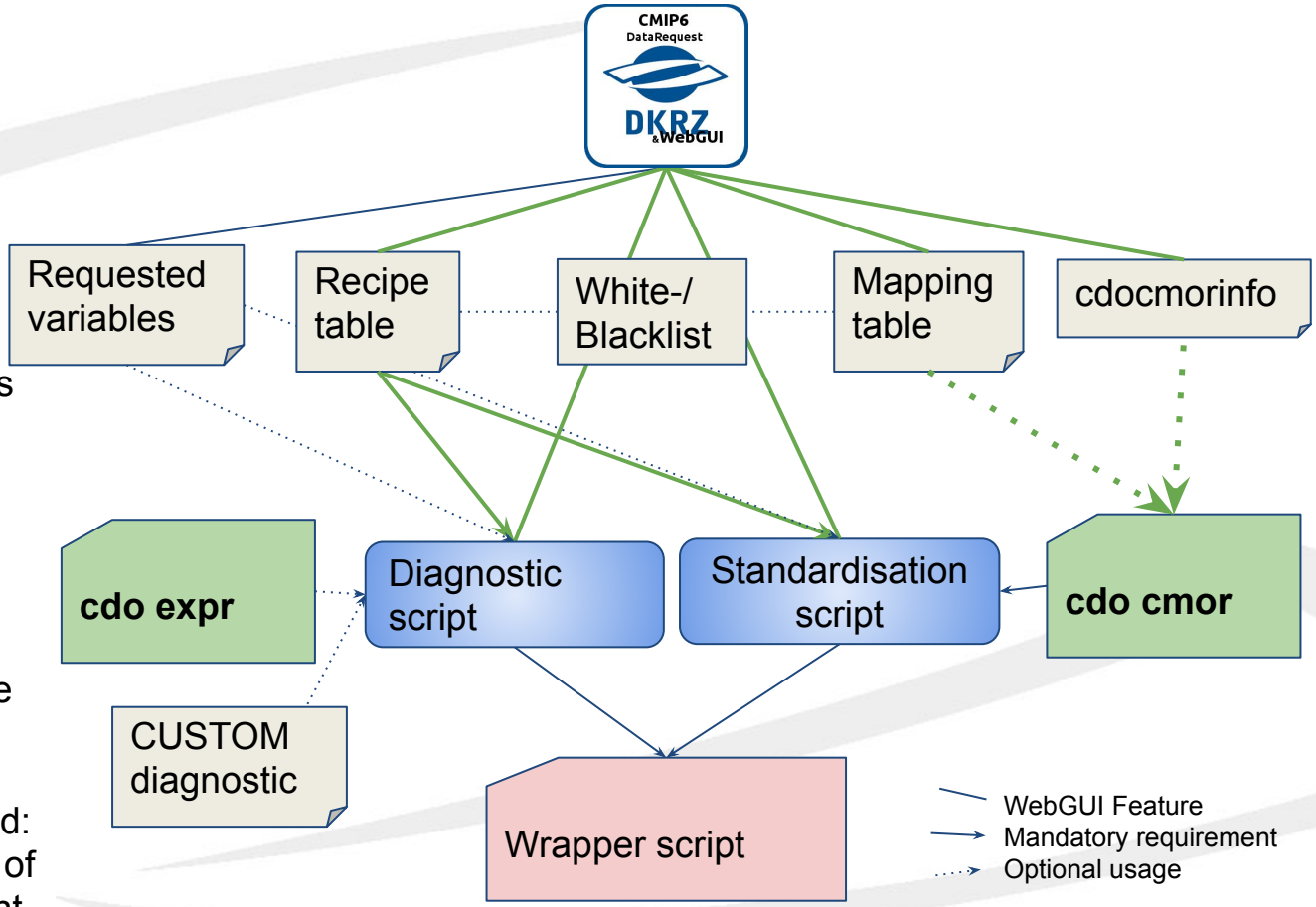
From WebGUI to runpp script

1. Fill in the tables for all CMOR variables which can be delivered
2. Generate script fragments
3. Generate list of global attributes for cdo cmor
4. Include script fragments in a post-processing script
5. There are methods to exclude variables:
   a. Requested variables configuration (one per experiment)
   b. White-/Blacklist
   c. Not recommended: remove parts out of the script fragment



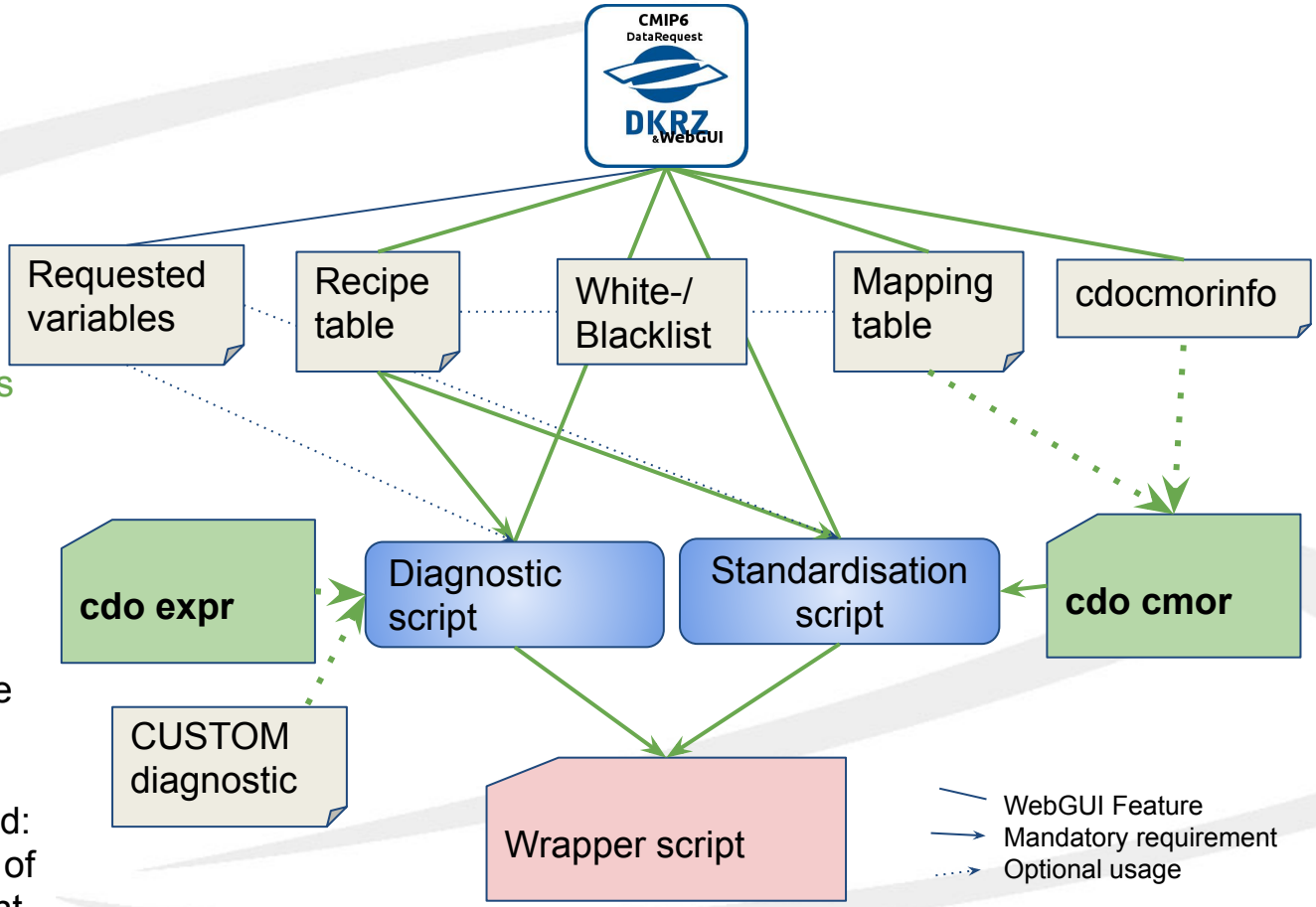
From WebGUI to runpp script

1. Fill in the tables for all CMOR variables which can be delivered
2. Generate script fragments
3. Generate list of global attributes for cdo cmor
4. Include script fragments in a post-processing script
5. There are methods to exclude variables:
   a. Requested variables configuration (one per experiment)
   b. White-/Blacklist
   c. Not recommended: remove parts out of the script fragment



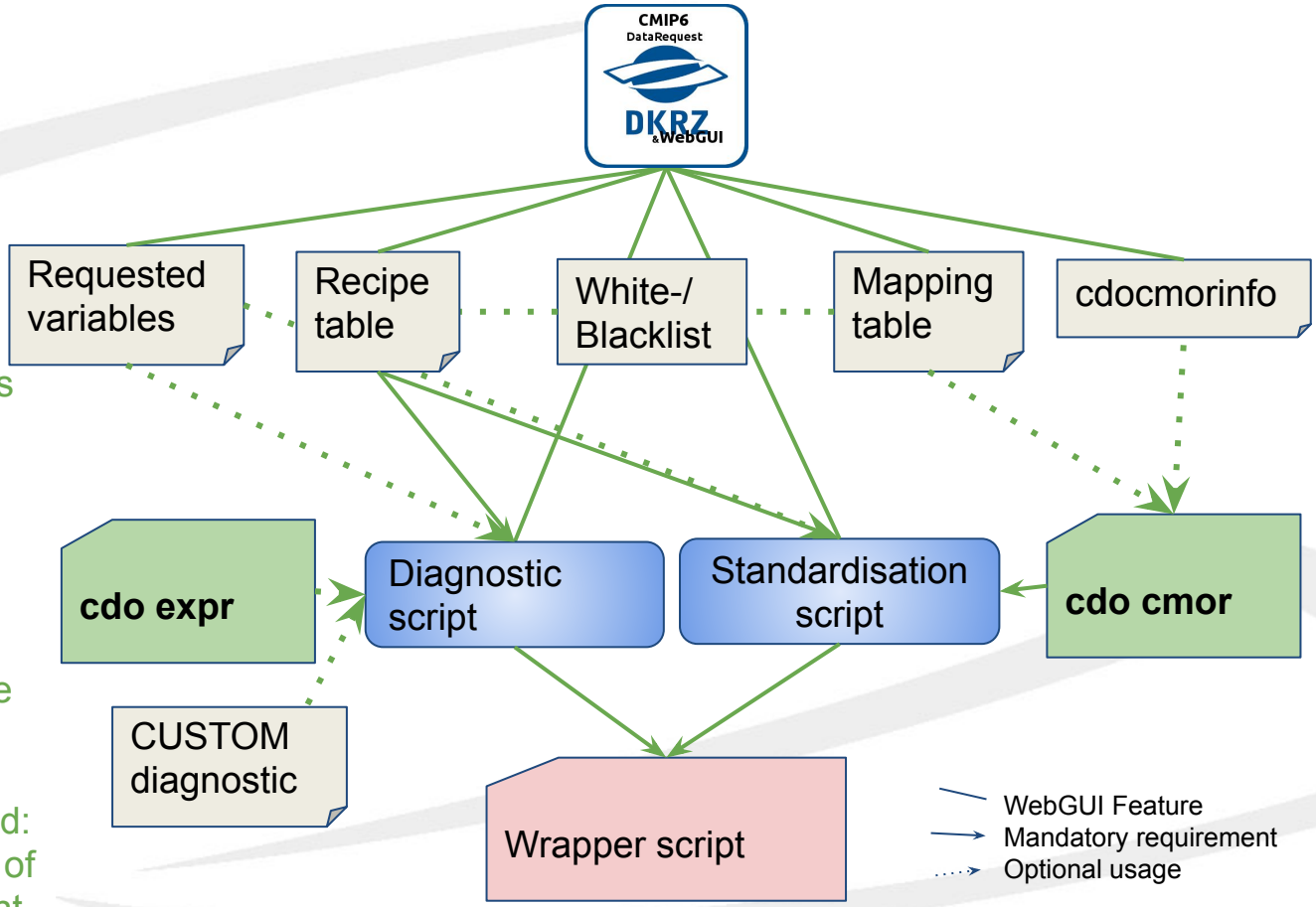
From WebGUI to runpp script

1. Fill in the tables for all CMOR variables which can be delivered
2. Generate script fragments
3. Generate list of global attributes for cdo cmor
4. Include script fragments in a post-processing script
5. There are methods to exclude variables:
   a. Requested variables configuration (one per experiment)
   b. White-/Blacklist
   c. Not recommended: remove parts out of the script fragment



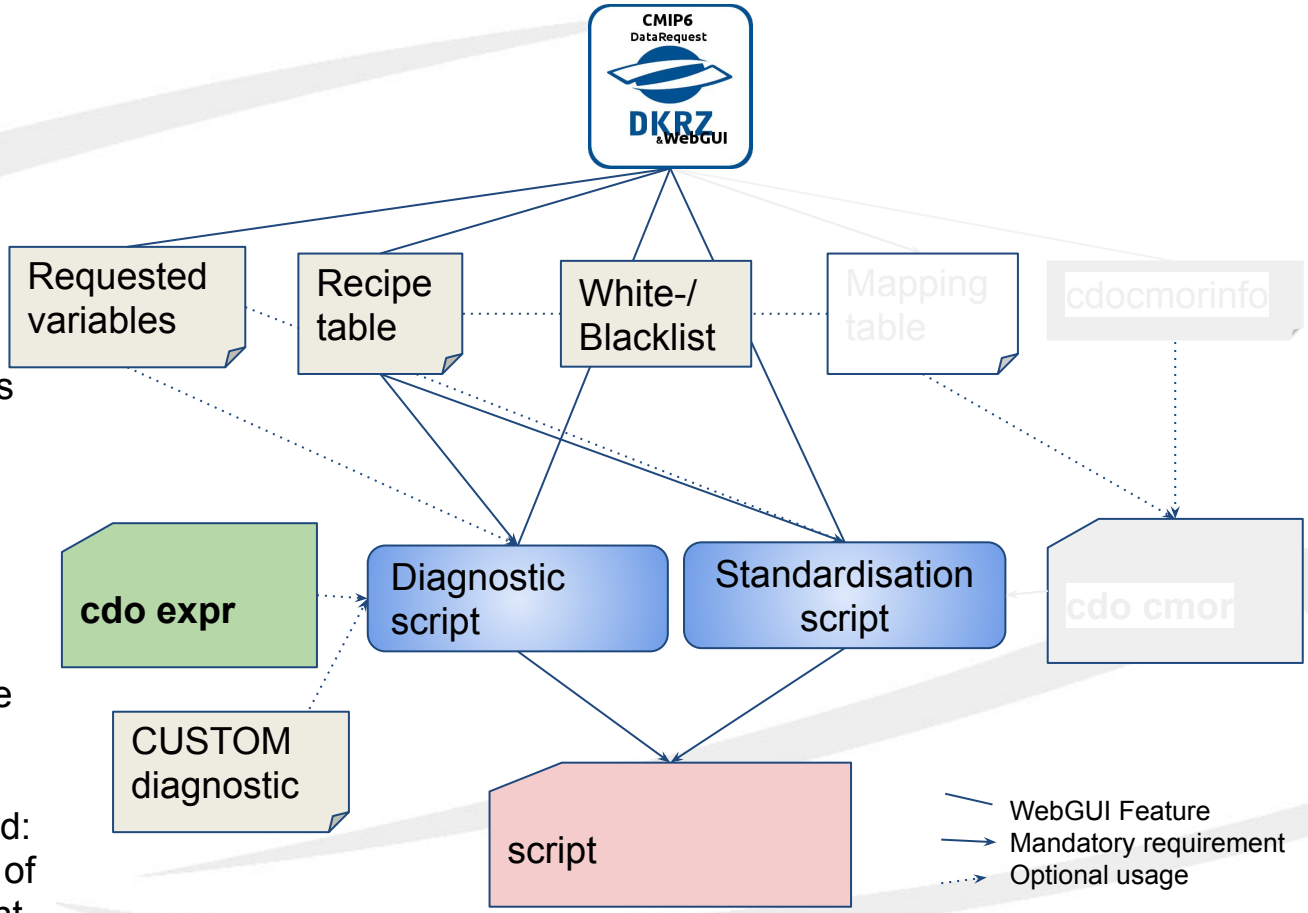
From WebGUI to runpp script

1. Fill in the tables for all CMOR variables which can be delivered
2. Generate script fragments
3. Generate list of global attributes for cdo cmor
4. Include script fragments in a post-processing script
5. There are methods to exclude variables:
   a. Requested variables configuration (one per experiment)
   b. White-/Blacklist
   c. Not recommended: remove parts out of the script fragment



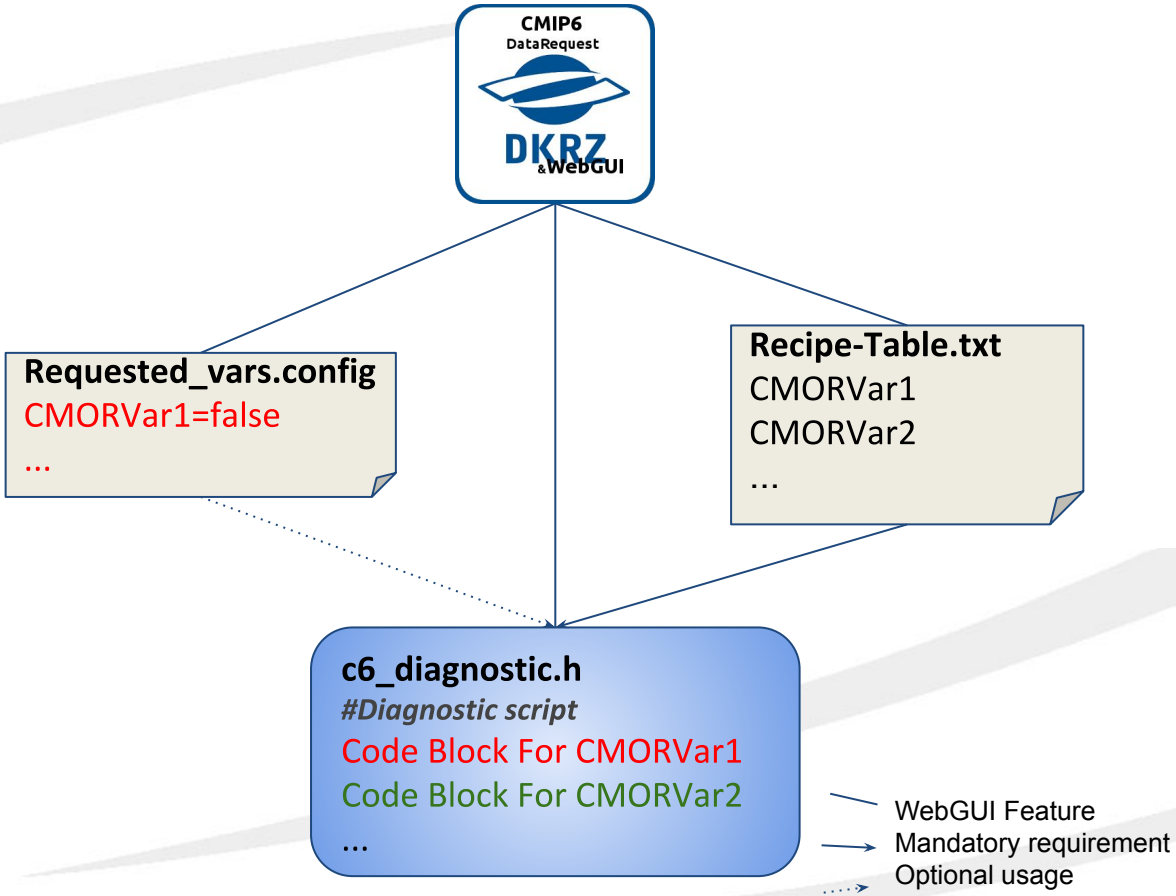
From WebGUI to runpp script

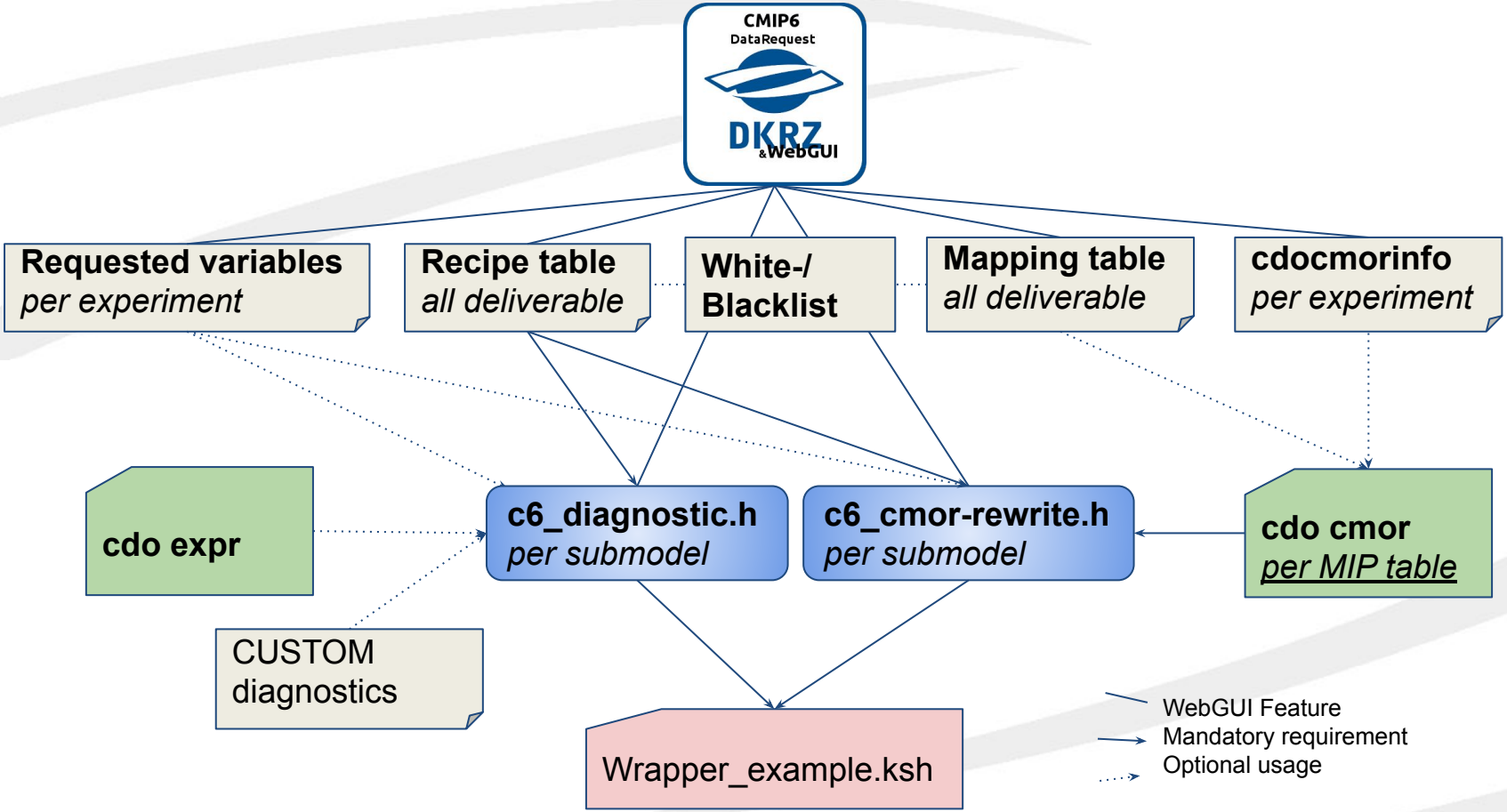
1. Fill in the tables for all CMOR variables which can be delivered
2. Generate script fragments
3. Generate list of global attributes for cdo cmor
4. Include script fragments in a post-processing script
5. There are methods to exclude variables:
   a. Requested variables configuration (one per experiment)
   b. White-/Blacklist
   c. Not recommended: remove parts out of the script fragment



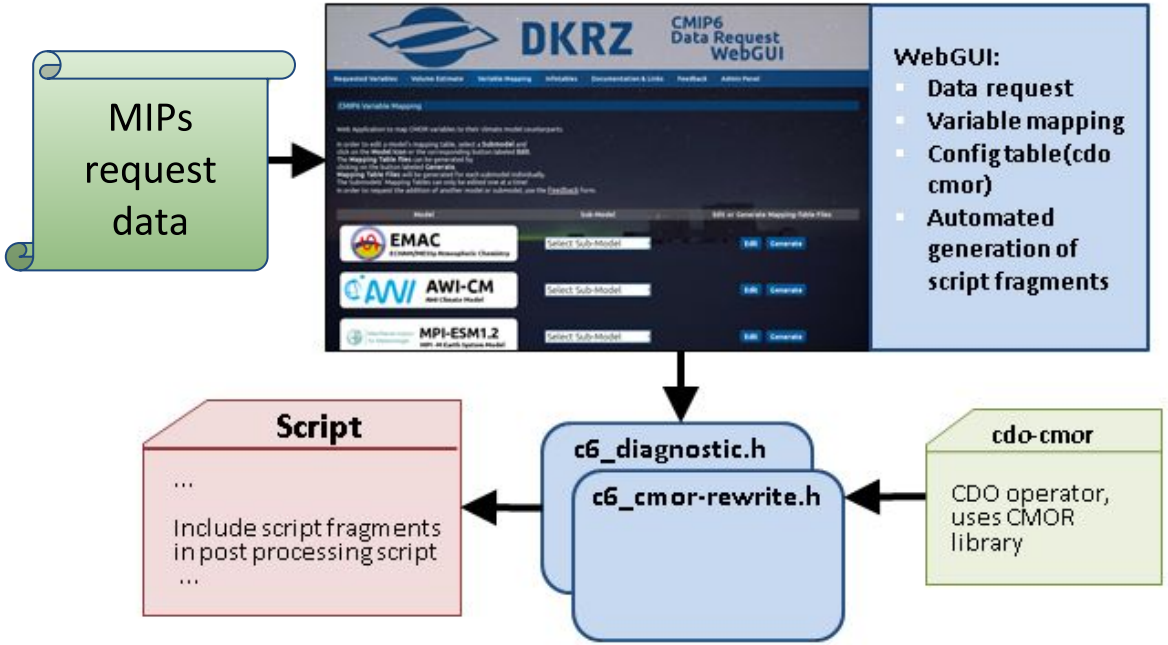
From WebGUI to runpp script - example

1. Fill in the tables for all CMOR variables which can be delivered
2. Generate script fragments
3. Generate list of global attributes for cdo cmor
4. Include script fragments in a post-processing script
5. There are methods to exclude variables:
   a. Requested variables configuration (one per experiment)
   b. White-/Blacklist
   c. Not recommended: remove parts out of the script fragment




CMIP6
DataRequest
DKRZ
&WebGUI
Requested variables
per experiment
Recipe table
all deliverable
White-/
Blacklist
Mapping table
all deliverable
cdocmorinfo
per experiment
cdo expr
c6_diagnostic.h
per submodel
c6_cmor-rewrite.h
per submodel
cdo cmor
per MIP table
CUSTOM
diagnostics
Wrapper_example.ksh
WebGUI Feature
Mandatory requirement
Optional usage

The Workflow - in brief



Workshop on CMIP6 Post-processing

cdo cmor - Introduction and first steps

Please see: https://slides.com/wachsylon/cdo-cmor-handson#/

Martin Schupfner (schupfner@dkrz.de),
Fabian Wachsmann (wachsmann@dkrz.de),
DKRZ

Workshop on CMIP6 Post-processing

Variable Mapping

Martin Schupfner (schupfner@dkrz.de),
Fabian Wachsmann (wachsmann@dkrz.de),
DKRZ

Exercise 1 - Preparation

1) Login to mistral and reserve a node using salloc:
salloc --nodes 1 --partition=compute,compute2 -A <project> -t 08:00:00 \ --mem=128G --
/bin/bash -c 'ssh -X $SLURM_JOB_NODELIST' [--qos=<qos>]

Alternatively to reserving a node you can later submit the scripts using "sbatch" instead of running it interactively on the reserved node.

On 9th and 10th October you need to use bk0988 as <project> and you can use "--qos=training" for a higher priority of your allocation request. Only the provided temporary training accounts can use the QOS!

2) Copy the directory **/work/bm0021/workshopcmip6pp2019/to_copy/testsim** to your work or scratch directory. This already includes the aggregated model output. No further aggregation is necessary.

cp -r /work/bm0021/workshopcmip6pp2019/to_copy/testsim /scratch/k/k2123456

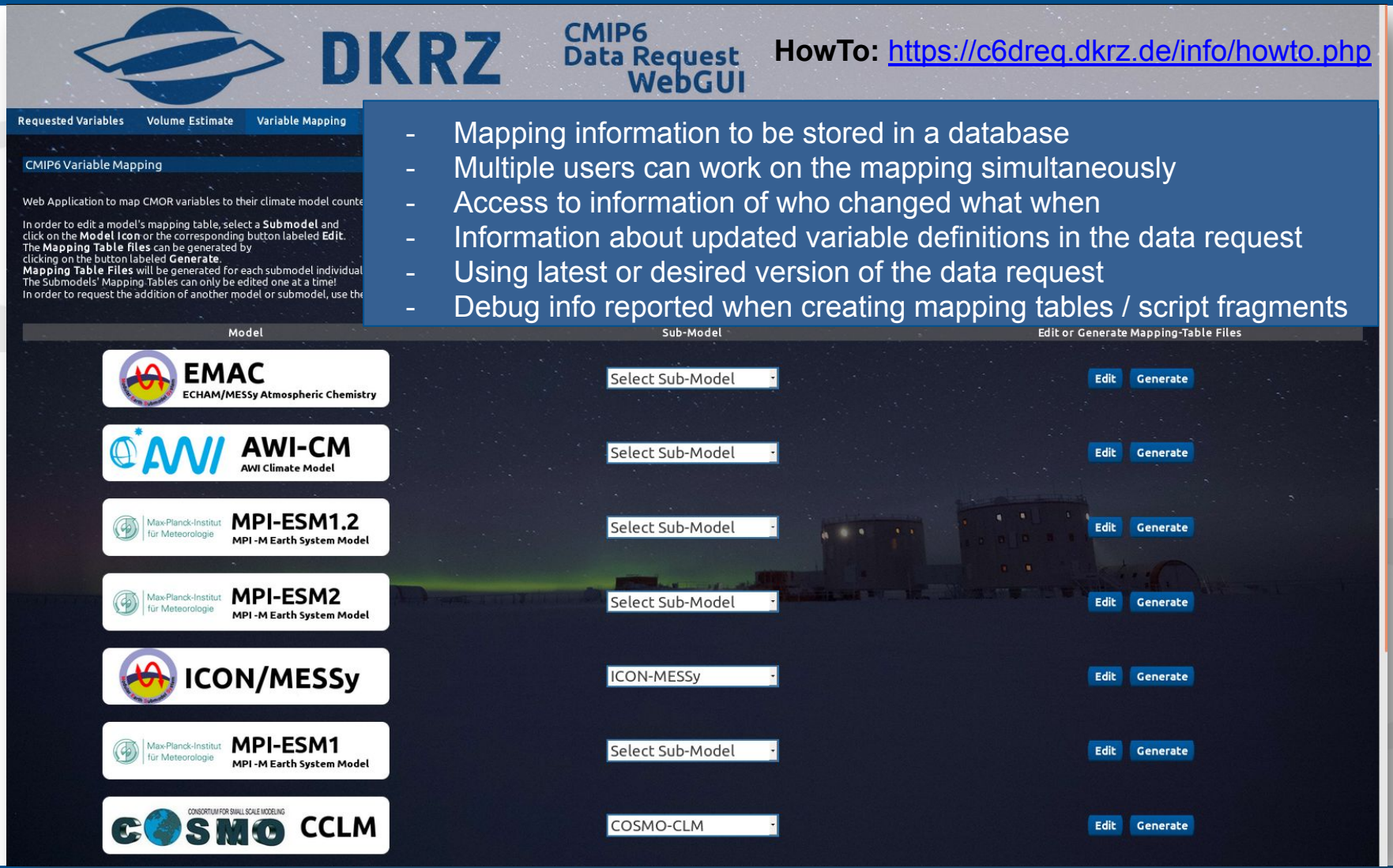
Variable Mapping Table

Map Model Variable to CMOR Variable

Model Output ⟵ **Data Request**

	CMOR variable name & MIP table		Model variable name / code	Units & positive flux direction	Requested cell methods	
&parameter	pmt=Amon	cmor_name=tasmax	c=201	units="K"	cell_methods="m"	/
&parameter	pmt=day	cmor_name=tasmax	c=201	units="K"	cell_methods="m"	/
&parameter	pmt=day	cmor_name=tasmin	c=202	units="K"	cell_methods="m"	/
&parameter	pmt=Amon	cmor_name=tasmin	c=202	units="K"	cell_methods="m"	/

optional: Variable comment attribute


DKRZ
CMIP6 Data Request WebGUI
HowTo: https://c6dreq.dkrz.de/info/howto.php
Requested Variables
Volume Estimate
Variable Mapping
CMIP6 Variable Mapping
Web Application to map CMOR variables to their climate model counte
In order to edit a model's mapping table, select a Submodel and click on the Model Icon or the corresponding button labeled Edit.
The Mapping Table files can be generated by clicking on the button labeled Generate.
Mapping Table Files will be generated for each submodel individual
The Submodels' Mapping Tables can only be edited one at a time!
In order to request the addition of another model or submodel, use the
- Mapping information to be stored in a database
- Multiple users can work on the mapping simultaneously
- Access to information of who changed what when
- Information about updated variable definitions in the data request
- Using latest or desired version of the data request
- Debug info reported when creating mapping tables / script fragments
Model
Sub-Model
Edit or Generate Mapping-Table Files
EMAC
ECHAM/MESSy Atmospheric Chemistry
Select Sub-Model
Edit
Generate
AWI-CM
AWI Climate Model
Select Sub-Model
Edit
Generate
MPI-ESM1.2
MPI -M Earth System Model
Select Sub-Model
Edit
Generate
MPI-ESM2
MPI -M Earth System Model
Select Sub-Model
Edit
Generate
ICON/MESSy
ICON-MESSy
Edit
Generate
MPI-ESM1
MPI -M Earth System Model
Select Sub-Model
Edit
Generate
CCLM
COSMO-CLM
Edit
Generate

Variable Mapping Information - Format

Important:

- Input files:
  - output from aggregation
  - best practise: naming convention within the submodel user group:
    e.g. “echamday-plev7h_1950.nc” for daily mean echam output on 7 pressure levels
- Recipes:
  - **cdo expr** operator
  - enter “CUSTOM” if **cdo expr** cannot be used
- Units/Positive:
  - Enter the units / positive flux direction from the aggregated/diagnosed variable
  - Inverting the flux direction or converting the units are NOT a diagnostic
  - **CMOR/cdo cmor** will do that automatically. This reduces I/O significantly

Diagnostics
The final script fragment contains *blocks* for each CMOR variable which is marked as “available” and has a recipe defined in the mapping table.

Standardisation
The final script fragment contains blocks for each CMOR-table and input file for all variables that are marked as “available”.

Variable Mapping Information - Format

Availability	
Model Variable Name	Filename1.nc|Varname1,Varname2;Filename2.nc|Varname3,Varname4
Model Variable Code	Filename1.grb|100,101;Filename2.grb|3,4
Post-Processing Recipe	toz=
Model Variable Units	m

Availability:

Selection menu - select available or unavailable, depending on the availability of that variable in the (sub)model

Variable Mapping Information - Format

Availability	
Model Variable Name	Filename1.nc|Varname1,Varname2;Filename2.nc|Varname3,Varname4
Model Variable Code	Filename1.grb|100,101;Filename2.grb|3,4
Post-Processing Recipe	toz=
Model Variable Units	m

Model Variable Name/Code:

Enter file information/pattern and the variable name/code in a certain format (state after the model output aggregation)

Variable Mapping Information - Format

Model Variable Name/Code - Example:

Output for January 1950 from different experiments, simulations & submodels:

historical_r1i1p1-HR_hamocc_data_3d_dm_19500101_19501231.nc

historical_r2i1p1-HR_hamocc_data_3d_dm_19500101_19501231.nc

piControl_r1i1p1-HR_hamocc_data_2d_mm_19500101-19501231.nc

experiment - placeholder: **EXPT**

Pattern:

EXPT_

Variable Mapping Information - Format

Model Variable Name/Code - Example:

Output for January 1950 from different experiments, simulations & submodels:

historical_r1i1p1-HR_hamocc_data_3d_dm_19500101_19501231.nc

historical_r2i1p1-HR_hamocc_data_3d_dm_19500101_19501231.nc

piControl_r1i1p1-HR_hamocc_data_2d_mm_19500101-19501231.nc

simulation/realisation - placeholder: **SIMU**

Pattern:

EXPT_SIMU_

Variable Mapping Information - Format

Model Variable Name/Code - Example:

Output for January 1950 from different experiments, simulations & submodels:

historical_r1i1p1-HR_hamocc_data_3d_dm_19500101_19501231.nc

historical_r2i1p1-HR_hamocc_data_3d_dm_19500101_19501231.nc

piControl_r1i1p1-HR_hamocc_data_2d_mm_19500101-19501231.nc

model - placeholder: **MODX**

Pattern:

EXPT_SIMU_MODX_

Variable Mapping Information - Format

Model Variable Name/Code - Example:

Output for January 1950 from different experiments, simulations & submodels:

historical_r1i1p1-HR_hamocc_data_3d_dm_19500101_19501231.nc

historical_r2i1p1-HR_hamocc_data_3d_dm_19500101_19501231.nc

piControl_r1i1p1-HR_hamocc_data_2d_mm_19500101-19501231.nc

unambiguous part(s) of the input file name, specifying eg. the output channel and aggregation

Pattern:

EXPT_SIMU_MODX_data_3d_dm_

Variable Mapping Information - Format

Model Variable Name/Code - Example:

Output for January 1950 from different experiments, simulations & submodels:

historical_r1i1p1-HR_hamocc_data_3d_dm_19500101_19501231.nc

historical_r2i1p1-HR_hamocc_data_3d_dm_19500101_19501231.nc

piControl_r1i1p1-HR_hamocc_data_2d_mm_19500101-19501231.nc

date string - placeholder: **DATE**

Pattern:

EXPT_SIMU_MODX_data_3d_dm_DATE*

Variable Mapping Information - Format

Model Variable Name/Code - Example:

Output for January 1950 from different experiments, simulations & submodels:

historical_r1i1p1-HR_hamocc_data_3d_dm_19500101_19501231.nc

historical_r2i1p1-HR_hamocc_data_3d_dm_19500101_19501231.nc

piControl_r1i1p1-HR_hamocc_data_2d_mm_19500101-19501231.nc

filename extension

Pattern:

EXPT_SIMU_MODX_data_3d_dm_DATE*.nc

Variable Mapping Information - Format

Model Variable Name/Code - Example:

Pattern:

EXPT_SIMU_MODX_data_3d_dm_DATE*.nc

Wildcards:

***** - substitutes any string, also an empty string
? - substitutes one character

Placeholders:

EXPT - substitutes the experiment
SIMU - substitutes the simulation/realisation
MODX - substitutes the (sub)model name
DATE - substitutes the date string

Requirement:

The entered pattern has to be **unambiguous**!

Alternative patterns for this input filename:

*_data_3d_dm_????????_????????.nc
*_data_3d_dm_*.nc
_data_3d_dm_DATE

How the entered pattern is dealt with:

find "$sdir" -name "${pattern}" | grep ${period}

where

sdir - source directory to be specified in the script
pattern - the input filename pattern
period - the date string (DATE - $period)

-> Using no wildcards or using the DATE placeholder will cause **the additional grep command** to be omitted when searching the filename.

Variable Mapping Information - Format

Availability	
Model Variable Name	Filename1.nc|Varname1,Varname2;Filename2.nc|Varname3,Varname4
Model Variable Code	Filename1.grb|100,101;Filename2.grb|3,4
Post-Processing Recipe	toz=
Model Variable Units	m

Model Variable Name/Code:

Enter file information/pattern and the variable name/code in a certain format (state after the model output aggregation)

***pattern*of_Filename1*|var1**
***pattern*of_Filename1*|var1,var2**
***pattern*of_Filename1*|var1;*pattern_of_Filename2.nc|var2**

Variable Mapping Information - Format

Post-Processing Recipe:

Enter a diagnostic recipe that can be interpreted by the cdo operator ‘expr’, eg.:

var3+sellevidx(var4,1) or **cli*0.004*${factor}**

Enter **CUSTOM** if the diagnostic cannot be performed by cdo expr. In this case, no file pattern and variable name / code has to be given. After the diagnostic the variable name has to be the same as the CMOR variable name!

Variable Mapping Information - Format

Availability	
Model Variable Name	Filename1.nc|Varname1,Varname2;Filename2.nc|Varname3,Varname4
Model Variable Code	Filename1.grb|100,101;Filename2.grb|3,4
Post-Processing Recipe	toz=
Model Variable Units	m

Model Variable Units:

Enter the units of the variable (state after the model output aggregation and, if applicable, the diagnostic), eg.:

kg or **kg m-2 s-1** or **1** or **%**

Exercise 2 - Mapping

1) Use your browser to login to the WebGUI (https://c6dreq.dkrz.de) and select your model with submodel “Atm” in the “CMIP6 Workshop” project for editing.
2) Get familiar with the options and fill in the mapping information for ps-Amon and tas-Amon.
3) Collect the missing mapping information “by all means” and fill it into the mapping table of the corresponding submodel.

Exercise 2 - Mapping

Sub-model	Frequency	Aggregation	CMOR/ short name	Name/Code in Model	Diagnostic (formula)	Unique filename pattern of the input files	Model output units	Variable Comment	Special Setting
Oce	Monthly	Climatology	difvho			*_ocean_3d_mm_DATE*	m2 s-1	Reported on ocean half levels.	
Oce	Decadal	Mean	hfbasin	global_hfbasin, atlantic_hfbasin, indopacific_hfbasin		*_ocean_moc_mm_DATE*	W		Decadal bounds (???9), define character axis basin in cdocmorinfo
Oce	fx		deptho			*_ocean_fx.nc	m		
Oce	Annual	Mean	co3	co3		*_ocean_3d_ym_DATE*	mol m-3		Vertical Bounds (gridinfo)
Atm	Monthly	Mean	tas	167		*_atmos_2d_mm_DATE*	K		
Atm	Monthly	Mean	ps	134		*_atmos_2d_mm_DATE*	Pa		
Atm	Monthly	Mean	pr	142,143	var142+var143	*_atmos_2d_mm_DATE*	kg m-2 s-1		
Atm	Daily	Zonal Mean	va	132 out of cdo afterburner (requires codes 129, 152, 138, 155)		*_atmos_3d_dm_plev19_zon_DATE*	m s-1		Aggregation (cdo zonmean) and cdo after call (pl interpolation and va-calculation) needed.
Atm	Monthly	Mean	cli	154 (and 134/ps)	var154:ps=var134	*_atmos_3d_mm_DATE* (and *_atmos_2d_mm_DATE* for 134/ps)	kg kg-1		Surface Pressure needed as auxiliary coordinate.
Srf	Monthly	Mean	landCoverFrac		*CUSTOM*		%		
Srf	Monthly	Mean	mrsol	84	1000*var84	*_land_3d_mm_DATE*	kg m-2		

Workshop on CMIP6 Post-processing

cdo cmor - Advanced

Please see: https://slides.com/wachsylon/cdo-cmor-handson#/

Martin Schupfner (schupfner@dkrz.de),
Fabian Wachsmann (wachsmann@dkrz.de),
DKRZ

Workshop on CMIP6 Post-processing

Automated generation
of
post processing script fragments

Martin Schupfner (schupfner@dkrz.de) ,
Fabian Wachsmann (wachsmann@dkrz.de),
DKRZ


DKRZ
CMIP6
Data Request
WebGUI
Logout ms
HowTo:
https://c6dreq.dkrz.de/info/howto3.php
Project
Mapping-Tables
Data Request (optional)
Script Creation
Build Post-Processing Scripts Fragments
Instructions to automatically build post-processing (diagnostic, CMOR rewrite) script fragments out of the variable mapping tables:
(1) Select Project: Select the project for which the script fragments have to be generated.
(2) Generate Mapping-Tables: Select the Models/Submodels you want the script fragments to be generated for.
(3) Generate Data Request (optional): In case you want the processing of each variable to be dependent on the project's official data request, generate a customized data request.
(4) Initiate Scripts Creation: Submit your selected options by clicking the Create Script Templates button.
or display a list of previously created scripts:
Browse Scripts
Select Project
CMIP6
Select
Current Selection
CMIP6 (Climate Model Intercomparison Project Phase 6)

Script fragments

- Automatic creation of diagnostic and cmor rewrite script fragment out of recipe table
- Automatic creation of data request configuration out of recipe table and CMIP6 data request, further customizable by user

Diagnostic
c6_diagnostic.h

- **cdo merge** (of multiple inputfiles)
- **cdo expr**

```
#-- Diagnostic for echam6 (ESM: AWI-CM-1-0-HR) variable rlds / table 3hr
{ (if_requested $member $atmmod 3hr rlds $chunk && {
  find_file -e -p $period "$sdir" "TST_????01.01_echam.grb" ifile1
  find_file -e            "$sdir" "rlds_3hr_${period}*" ifile2
  $cdo -f nc -0 \
    expr,'rlds=var177-var205;' \
    -merge -selcode,177 $ifile1 -selcode,205 $ifile2 \
    in_cmor/3hr_rlds_$period.nc || echo ERROR
}; )&; }>$err.rlds.3hr 2>&1
```

CMOR rewrite
c6_cmor-rewrite.h

- **cdo cmor** call

```
#-- CMOR-rewrite for echam6 (ESM: AWI-CM-1-0-HR) 3hr
cn='rlds hfls'
for var in $cn; do
  { (if_requested $member $atmmod 3hr $var $chunk || continue
  ifile=in_cmor/3hr_${var}_$period.nc
  $cdo cmor,3hr,mt=$mt,dr=$dr,cn=$var $ifile || echo ERROR
  )&; }>>$err.$var.3hr 2>&1
done
```

The script fragment does ...

- … test if variable is requested (data request, timeslice, user specifications)
- … find inputfile & call cdo

Script fragments

- Automatic creation of diagnostic and cmor rewrite script fragment out of recipe table
- Automatic creation of data request configuration out of recipe table and CMIP6 data request, further customizable by user

Diagnostic
c6_diagnostic.h

- **cdo merge** (of multiple inputfiles)
- **cdo expr**

```
#-- Diagnostic for echam6 (ESM: AWI-CM-1-0-HR) variable rlds / table 3hr
{ (if_requested $member $atmmod 3hr rlds $chunk && {
  find_file -e -p $period "$sdir" "TST_????01.01_echam.grb" ifile1
  find_file -e            "$sdir" "rlds_3hr_${period}*" ifile2
  $cdo -f nc -0 \
    expr,'rlds=var177-var205;' \
    -merge -selcode,177 $ifile1 -selcode,205 $ifile2 \
    in_cmor/3hr_rlds_$period.nc || echo ERROR
  )&; }>>$err.rlds.3hr 2>&1
```

CMOR rewrite
c6_cmor-rewrite.h

- **cdo cmor** call

```
#-- CMOR-rewrite for echam6 (ESM: AWI-CM-1-0-HR) 3hr
cn='rlds hfls'
for var in $cn; do
  { (if_requested $member $atmmod 3hr $var $chunk || continue
  ifile=in_cmor/3hr_${var}_$period.nc
  $cdo cmor,3hr,mt=$mt,dr=$dr,cn=$var $ifile || echo ERROR
  )&; }>>$err.$var.3hr 2>&1
done
```

The script fragment does ...

- … test if variable is requested (data request, timeslice, user specifications)
- … find inputfile & call cdo

Script fragments

- Automatic creation of diagnostic and cmor rewrite script fragment out of recipe table
- Automatic creation of data request configuration out of recipe table and CMIP6 data request, further customizable by user

Diagnostic
c6_diagnostic.h

- **cdo merge** (of multiple inputfiles)
- **cdo expr**

```
#-- Diagnostic for echam6 (ESM: AWI-CM-1-0-HR) variable rlds / table 3hr
{ (if_requested $member $atmmod 3hr rlds $chunk && {
  find_file -e -p $period "$sdir" "TST_??????01.01_echam.grb" ifile1
  find_file -e            "$sdir" "rlds_3hr_${period}*" ifile2
  $cdo -f nc -0 \
    expr,'rlds=var177-var205;' \
    -merge -selcode,177 $ifile1 -selcode,205 $ifile2 \
    in_cmor/3hr_rlds_$period.nc || echo ERROR
}; )&; }>$err.rlds.3hr 2>&1
```

CMOR rewrite
c6_cmor-rewrite.h

- **cdo cmor** call

```
#-- CMOR-rewrite for echam6 (ESM: AWI-CM-1-0-HR) 3hr
cn='rlds hfls'
for var in $cn; do
  { (if_requested $member $atmmod 3hr $var $chunk || continue
  ifile=in_cmor/3hr_${var}_$period.nc
  $cdo cmor,3hr,mt=$mt,dr=$dr,cn=$var $ifile || echo ERROR
  )&; }>>$err.$var.3hr 2>&1
done
```

The script fragment does ...

- … test if variable is requested (data request, timeslice, user specifications)
- … find inputfile & call cdo

Script fragments

- Automatic creation of diagnostic and cmor rewrite script fragment out of recipe table
- Automatic creation of data request configuration out of recipe table and CMIP6 data request, further customizable by user

Diagnostic
c6_diagnostic.h
- **cdo merge** (of multiple inputfiles)
- **cdo expr**

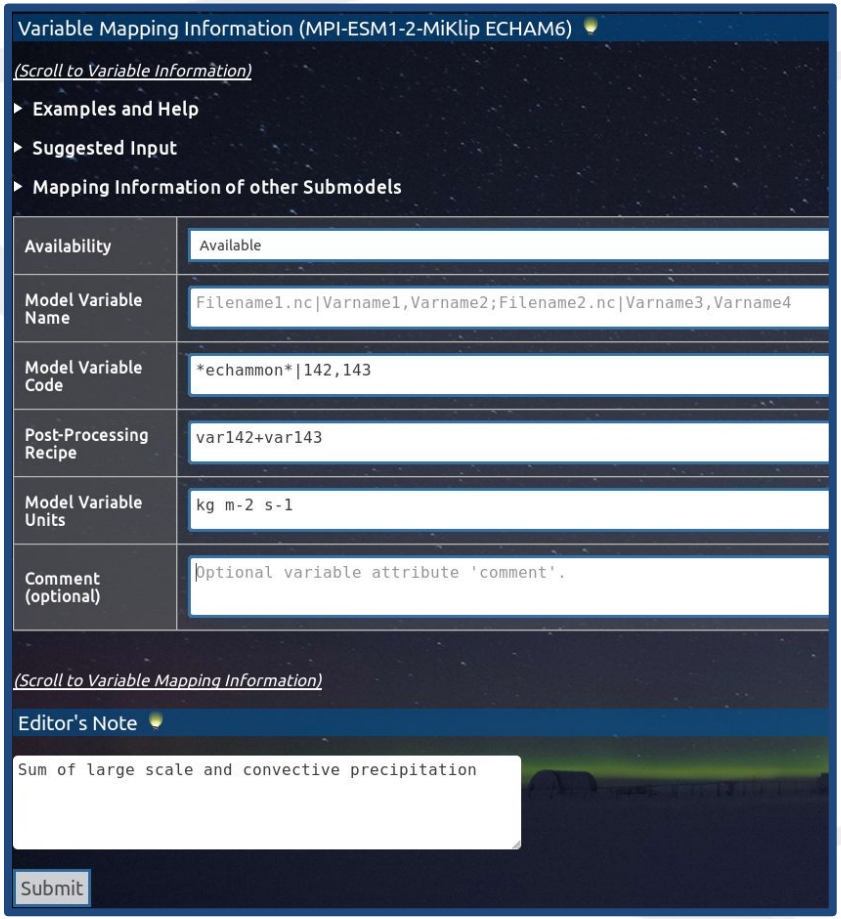
```
#-- Diagnostic for echam6 (ESM: AWI-CM-1-0-HR) variable rlds / table 3hr
{ (if_requested $member $atmmod 3hr rlds $chunk && {
  find_file -e -p $period "$sdir" "TST_????01.01_echam.grb" ifile1
  find_file -e            "$sdir" "rlds_3hr_${period}*" ifile2
  $cdo -f nc -0 \
    expr,'rlds=var177-var205;' \
    -merge -selcode,177 $ifile1 -selcode,205 $ifile2 \
    in_cmor/3hr_rlds_$period.nc || echo ERROR
}; )&; }>$err.rlds.3hr 2>&1
```

CMOR rewrite
c6_cmor-rewrite.h
- **cdo cmor** call

```
#-- CMOR-rewrite for echam6 (ESM: AWI-CM-1-0-HR) 3hr
cn='rlds hfls'
for var in $cn; do
  { (if_requested $member $atmmod 3hr $var $chunk || continue
  ifile=in_cmor/3hr_${var}_$period.nc
  $cdo cmor,3hr,mt=$mt,dr=$dr,cn=$var $ifile || echo ERROR
  )&; }>>$err.$var.3hr 2>&1
done
```

The script fragment does ...
- … test if variable is requested (data request, timeslice, user specifications)
- … find inputfile & call cdo

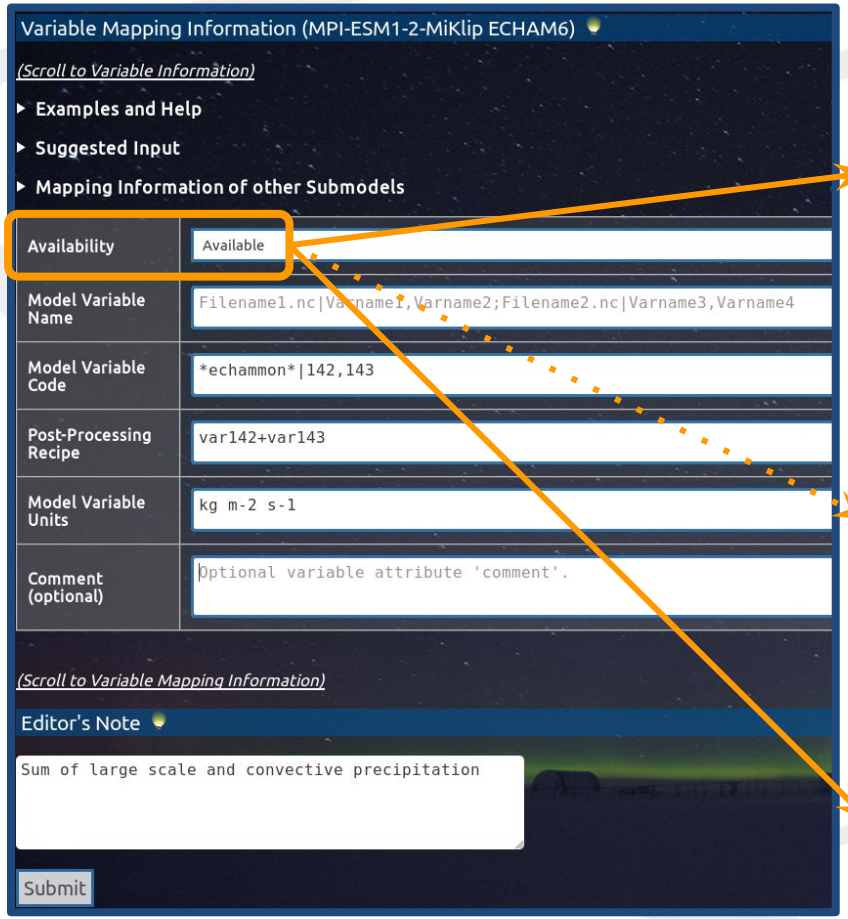
From Mapping-Table to Script Fragment



Example:

total precipitation
as the sum of large scale
and convective precipitation

From Mapping-Table to Script Fragment



Diagnostic script fragment

```
#-- Diagnostic for atmosphere (ESM: ESM) variable pr / table Amon
# Editor's note: Sum of large scale and convective precipitation
{ (if requested $member $atmmod Amon pr $chunk && {
  find file -e -p $period "$sdir" "*echammon*" ifile
  $cdo -f nc -0 \
    expr,'pr=var142+var143;' \
    $ifile in cmor/Amon_pr_$period.nc || echo ERROR
}; )&; }>$err.pr.Amon 2>&1
```

Mapping table

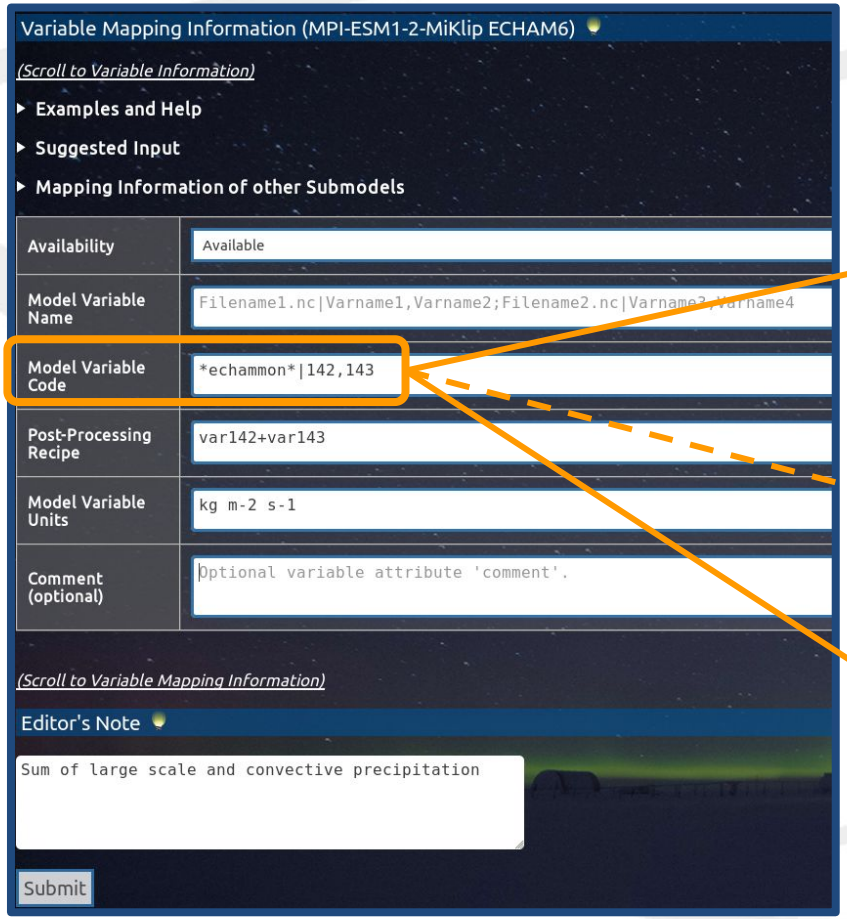
```
&parameter  pmt=Amon
            cmor_name=pr
            units="kg m-2 s-1"
            cell_methods="m"
            comment="" /
```

MT is a subset of the RT

Recipe table

```
#Editor's note: Sum of large scale and convective precipitation
&parameter  codes="*echammon*|142,143"
            recipe="pr=var142+var143;" /
```

From Mapping-Table to Script Fragment



Diagnostic script fragment

```
#-- Diagnostic for atmosphere (ESM: ESM) variable pr / table Amon
# Editor's note: Sum of large scale and convective precipitation
{ (if requested $member $atmmod Amon pr $chunk && {
  find file -e -p $period "$sdir" "*echammon*" ifile
  $cdo -f nc -0 \
    expr,'pr=var142+var143;' \
    $ifile in cmor/Amon_pr_$period.nc || echo ERROR
}; )&; }>$err.pr.Amon 2>&1
```

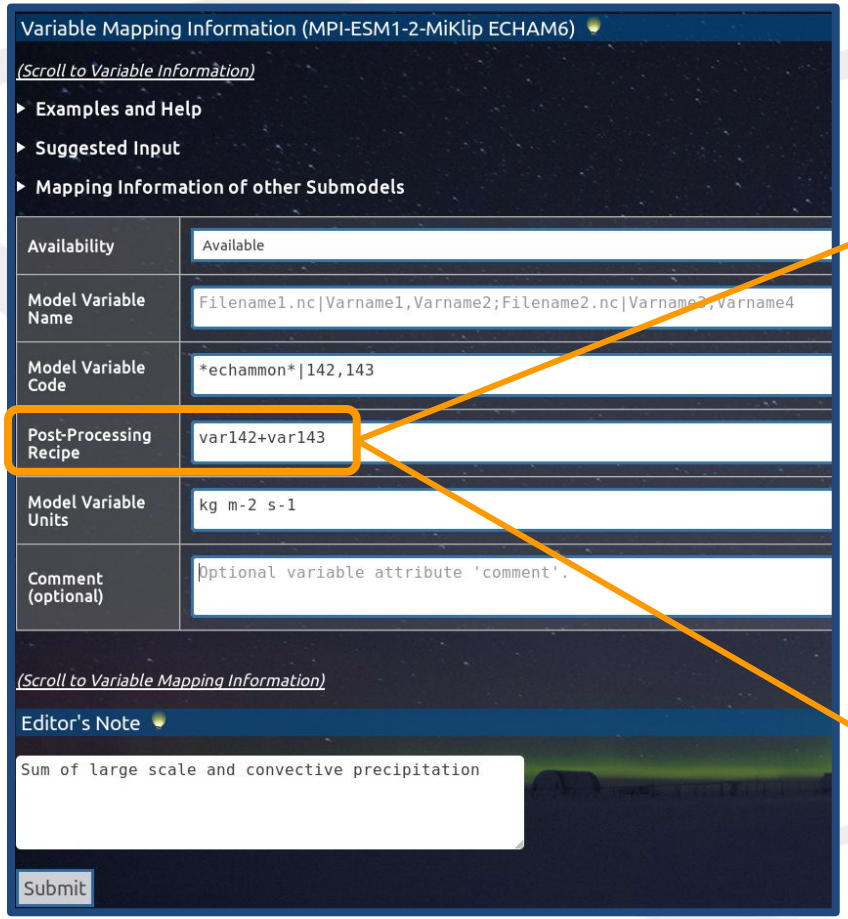
Mapping table

```
&parameter  pmt=Amon
            cmor_name=pr
            units="kg m-2 s-1"
            cell_methods="m"
            comment="" /
```

Recipe table

```
#Editor's note: Sum of large scale and convective precipitation
&parameter  codes="*echammon*|142,143"
            recipe="pr=var142+var143;" /
```

From Mapping-Table to Script Fragment



Diagnostic script fragment

```
#-- Diagnostic for atmosphere (ESM: ESM) variable pr / table Amon
# Editor's note: Sum of large scale and convective precipitation
{ (if requested $member $atmmod Amon pr $chunk && {
  find file -e -p $period "$sdir" "*echammon*" ifile
  $cdo -f nc -O \
    expr,'pr=var142+var143;' \
    $ifile in cmor/Amon_pr_$period.nc || echo ERROR
}; )&; }>$err.pr.Amon 2>&1
```

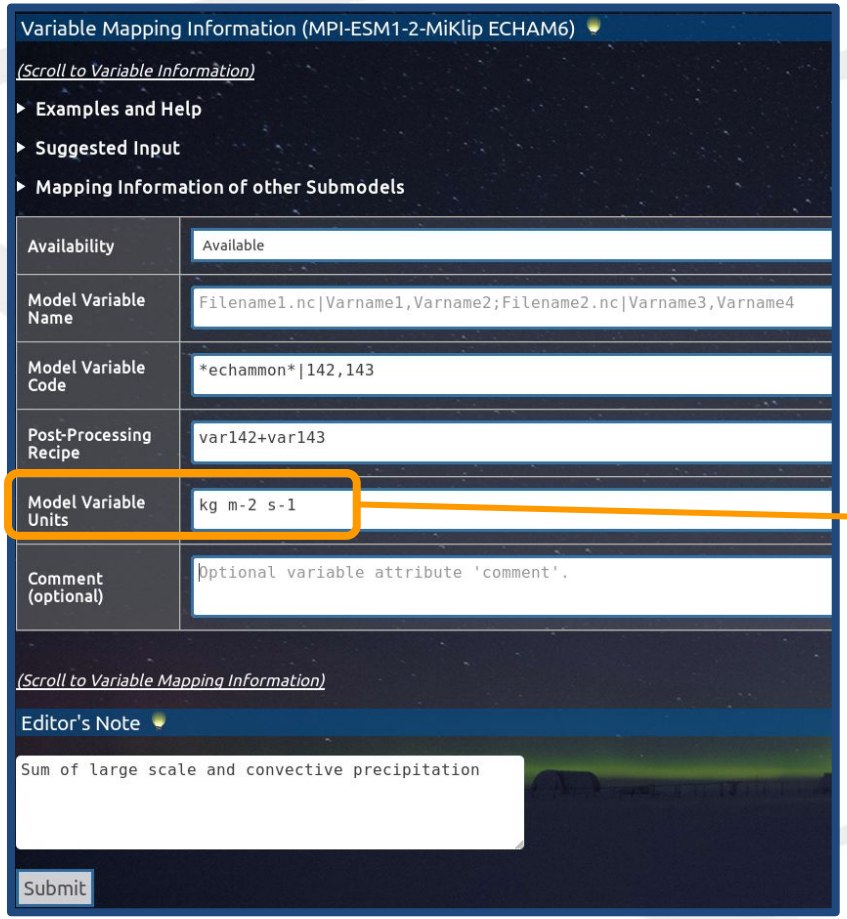
Mapping table

```
&parameter  pmt=Amon
            cmor_name=pr
            units="kg m-2 s-1"
            cell_methods="m"
            comment="" /
```

Recipe table

```
#Editor's note: Sum of large scale and convective precipitation
&parameter  codes="*echammon*|142,143"
            recipe="pr=var142+var143;" /
```

From Mapping-Table to Script Fragment



Diagnostic script fragment

```
#-- Diagnostic for atmosphere (ESM: ESM) variable pr / table Amon
# Editor's note: Sum of large scale and convective precipitation
{ (if requested $member $atmmod Amon pr $chunk && {
  find file -e -p $period "$sdir" "*echammon*" ifile
  $cdo -f nc -0 \
    expr,'pr=var142+var143;' \
    $ifile in cmor/Amon_pr_$period.nc || echo ERROR
}; )&; }>$err.pr.Amon 2>&1
```

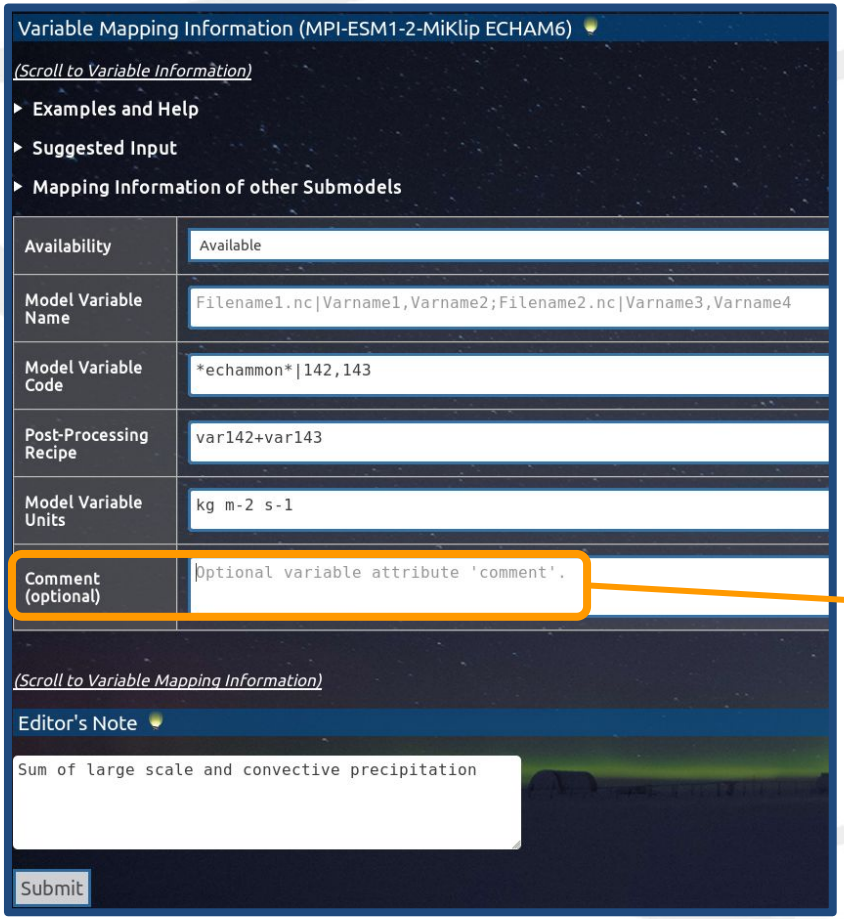
Mapping table

```
&parameter  pmt=Amon
            cmor_name=pr
            units="kg m-2 s-1"
            cell_methods="m"
            comment="" /
```

Recipe table

```
#Editor's note: Sum of large scale and convective precipitation
&parameter  codes="*echammon*|142,143"
            recipe="pr=var142+var143;" /
```

From Mapping-Table to Script Fragment



Diagnostic script fragment

```
#-- Diagnostic for atmosphere (ESM: ESM) variable pr / table Amon
# Editor's note: Sum of large scale and convective precipitation
{ (if requested $member $atmmod Amon pr $chunk && {
  find file -e -p $period "$sdir" "*echammon*" ifile
  $cdo -f nc -0 \
    expr,'pr=var142+var143;' \
    $ifile in cmor/Amon_pr_$period.nc || echo ERROR
}; )&; }>$err.pr.Amon 2>&1
```

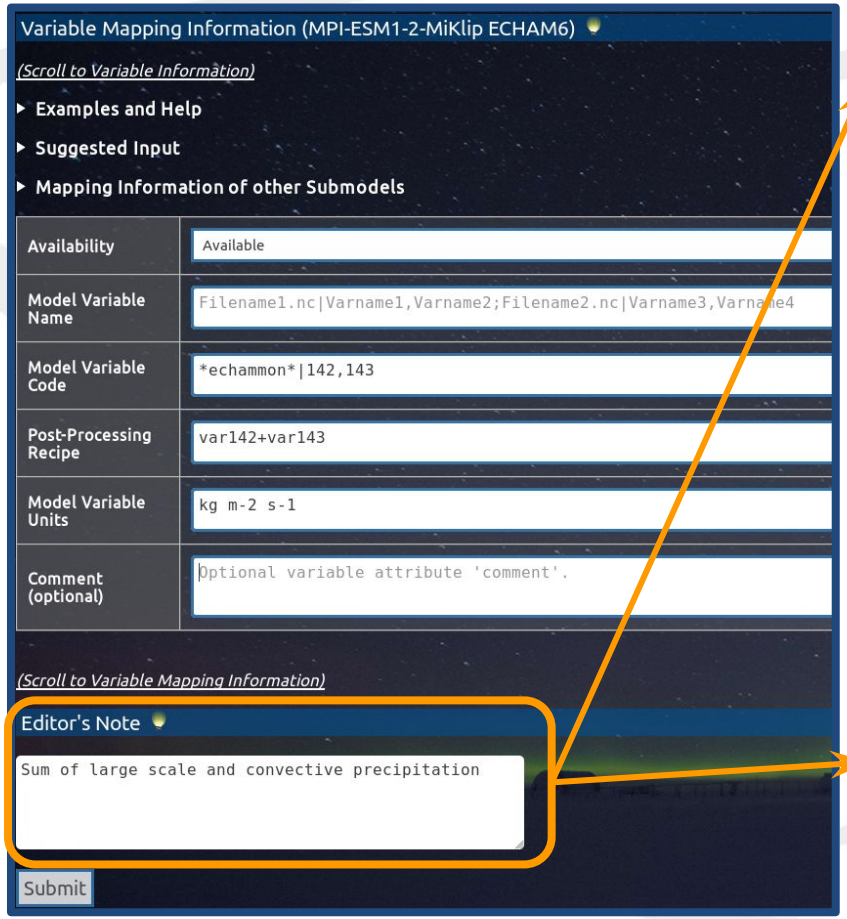
Mapping table

```
&parameter  pmt=Amon
            cmor_name=pr
            units=”kg m-2 s-1”
            cell_methods=”m”
            comment=”” /
```

Recipe table

```
#Editor’s note: Sum of large scale and convective precipitation
&parameter  codes=”*echammon*|142,143”
            recipe=”pr=var142+var143;” /
```

From Mapping-Table to Script Fragment



Diagnostic script fragment

```
#-- Diagnostic for atmosphere (ESM: ESM) variable pr / table Amon
# Editor's note: Sum of large scale and convective precipitation
{ (if requested $member $atmmod Amon pr $chunk && {
  find file -e -p $period "$sdir" "*echammon*" ifile
  $cdo -f nc -0 \
    expr,'pr=var142+var143;' \
    $ifile in cmor/Amon_pr_$period.nc || echo ERROR
}; )&; }>$err.pr.Amon 2>&1
```

Mapping table

```
&parameter  pmt=Amon
            cmor_name=pr
            units="kg m-2 s-1"
            cell_methods="m"
            comment="" /
```

Recipe table

```
#Editor's note: Sum of large scale and convective precipitation
&parameter  codes="*echammon*|142,143"
            recipe="pr=var142+var143;" /
```

Exercise 3 - Generate script fragments

1) Use your browser to login to the WebGUI (https://c6dreq.dkrz.de) and select the PostProcessing tab.
2) Create a script for the variables in the provided table. You can mask other variables using the Black-/Whitelist or you mask them later via the requested vars configuration.
3) Download and extract the script in your ./testsim/scripts directory.

The configuration files

requested_vars.conf
CMORVar1=false
MIPtableXY=false
CMORVar2=true
…

CMIP6_Amon.json

contains parts of the data request in a CMOR-readable format

```
if_requested amip CMORvar … && {
cdo cmor,CMIP6_Amon.json\
    gi=grid_info.nc,\
     i=.cdocmorinfo,\
    mt=mapping_table.txt\
infile }
```

grid_info.nc
contains a grid description including coordinates and bounds

```
variables:
    double lat(lat);
    double
lat_bnds(lat,bnds);
```

.cdocmorinfo
contains the CMOR configuration

```
activity_id="CMIP"
```

can be created with https://c6dreq.dkrz.de/cdocmorinfo/index.html

mapping_table.txt
links model output variables with CMOR variables

```
&parameter pmt=Amon cmor_name=tasmax  code=201
/
```

Data Request Config - *CMIP6_requested_vars_historical.conf* *one file per experiment*

```
################################################
EXP=historical
################################################

DREQSETTINGS
Emon        : Emon       = slice: piControl030,piControl050
Emon        : thetaot300 = False
EmonZ       : EmonZ      = False
Amon        : no2        = False

USERSETTINGS
# ---> Specify your settings for Experiment historical here
Emon        : hus        = slice: 1900010100-1924123124
Emon        : co2s       = slice: TOTAL
3hr         : 3hr        = False
Lmon        : jsbach     = False
E1hr        : r2i1p1f1   = False
# <---- Specify your settings for Experiment historical here
```

Format

Standard:
Field:Key = True/False
Field:Key = slice:<slicename>
Field:Key = slice:<slicename1>,<slicename2>,...
Field:Key=slice:YYYYMMDDHH-YYYYMMDDHH

Usersettings are given priority. But User and DReq timeslices are added.

Advanced:
Shell-Variable:
Field:Key = switch
with $switch being a shell variable defined as True/False

Option:
Option:<optname>=true/false
-> with <optname> being a “SettingContainer” defined in the script:

- a collection of standard settings that can be activated or deactivated altogether, eg. all settings for an aquaplanet or active/inactive dynamic vegetation

Field/Key can be: CMOR variable, MIPtable, submodel, experiment realisation and TOTAL (High to low priority).
if_requested call: All settings will be scanned from high to low priority. First matching setting will be applied.

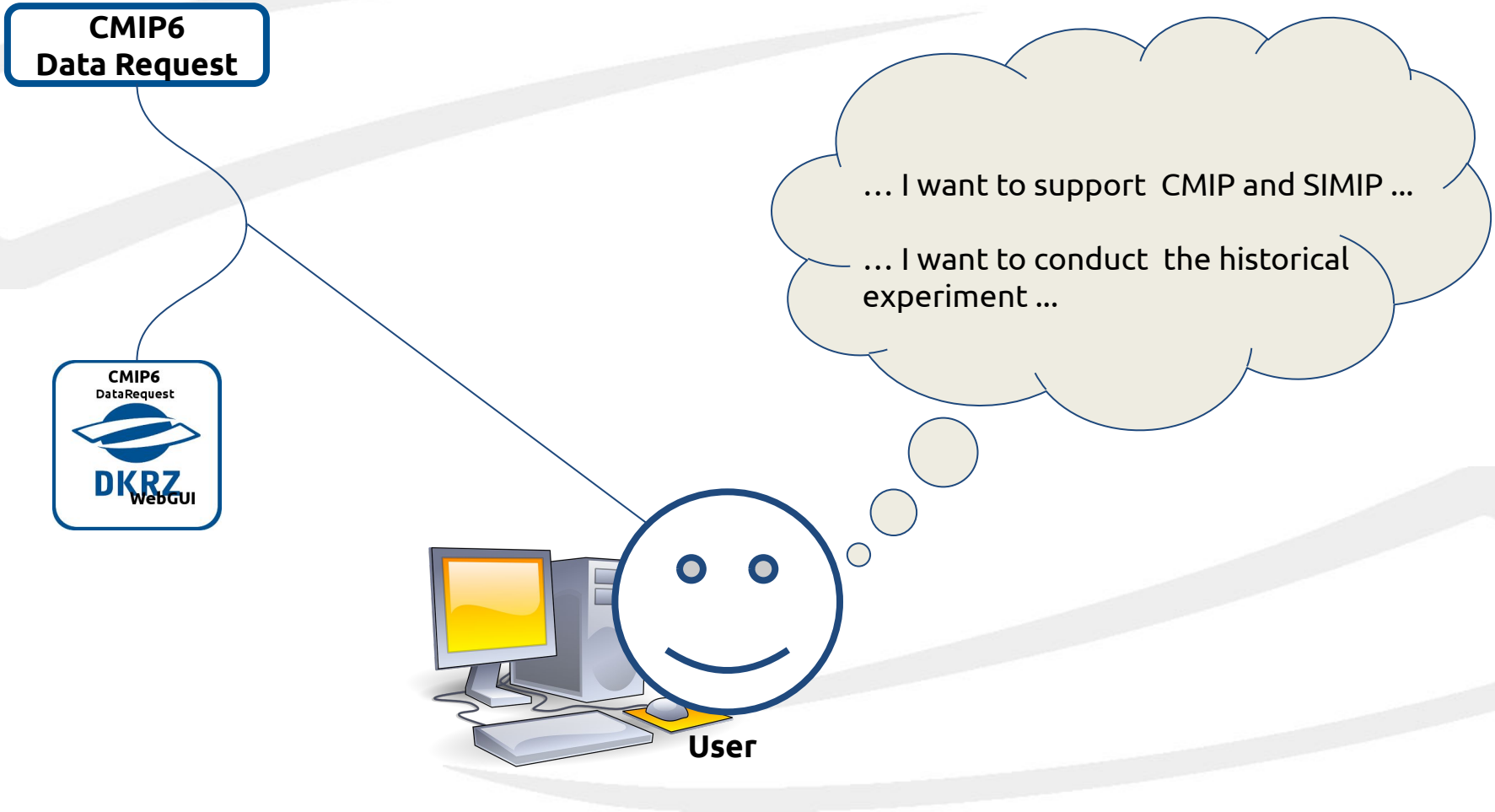
Exercise 4 - Execute the diagnostic and standardizing scripts

1) Take a look into the scripts and perform all necessary changes.
   - *./conf/*requested_vars*.conf:*
     In the requested variables configuration file (./conf/), add a timeslice for Oclim variables for 1850 to 1859. Set the processing for variables to “False” that are not in the output (if you did not use the Black-/Whitelist when creating the script fragments).
   - *Wrapper_example.ksh: path and variables definitions, settings*
   - *cdocmorinfo: cdo cmor settings, CMOR global attributes*
2) Run the script. As you reserved a node and we just deal with a few test variables, you can run it interactively:

   ksh Wrapper_example.ksh

   Alternatively submit:

   sbatch Wrapper_example.ksh


CMIP6
Data Request
CMIP6
DataRequest
DKRZ
WebGUI
… I want to support CMIP and SIMIP …
… I want to conduct the historical experiment …
User

Variable Mapping

mapping_table.txt
links model output variables with CMOR variables

```
&parameter pmt=Amon
cmor_name=tasmax  code=201 /
```

CMIP6 DataRequest WebGUI

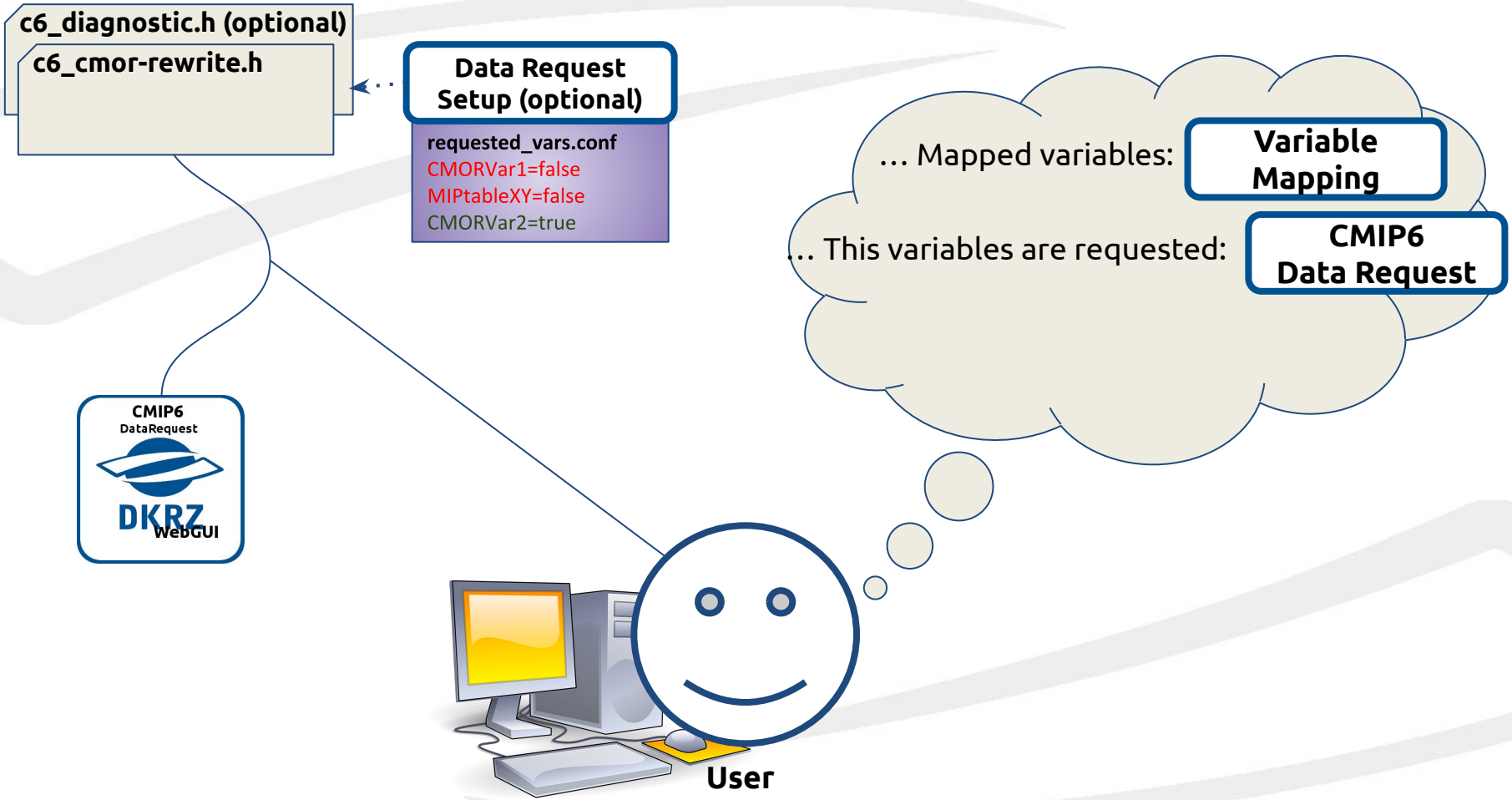
… I run the model MPI-ESM-1-2-HR…

… These variables are requested:

CMIP6 Data Request

… Available model output and diagnostic …

User


c6_diagnostic.h (optional)
c6_cmor-rewrite.h
Data Request Setup (optional)
requested_vars.conf
CMORVar1=false
MIPtableXY=false
CMORVar2=true
… Mapped variables:
Variable Mapping
… This variables are requested:
CMIP6 Data Request
CMIP6 DataRequest
DKRZ WebGUI
User

CDO CMOR Config Table

.cdocmorinfo

contains the user configuration

```
activity_id="CMIP"
```

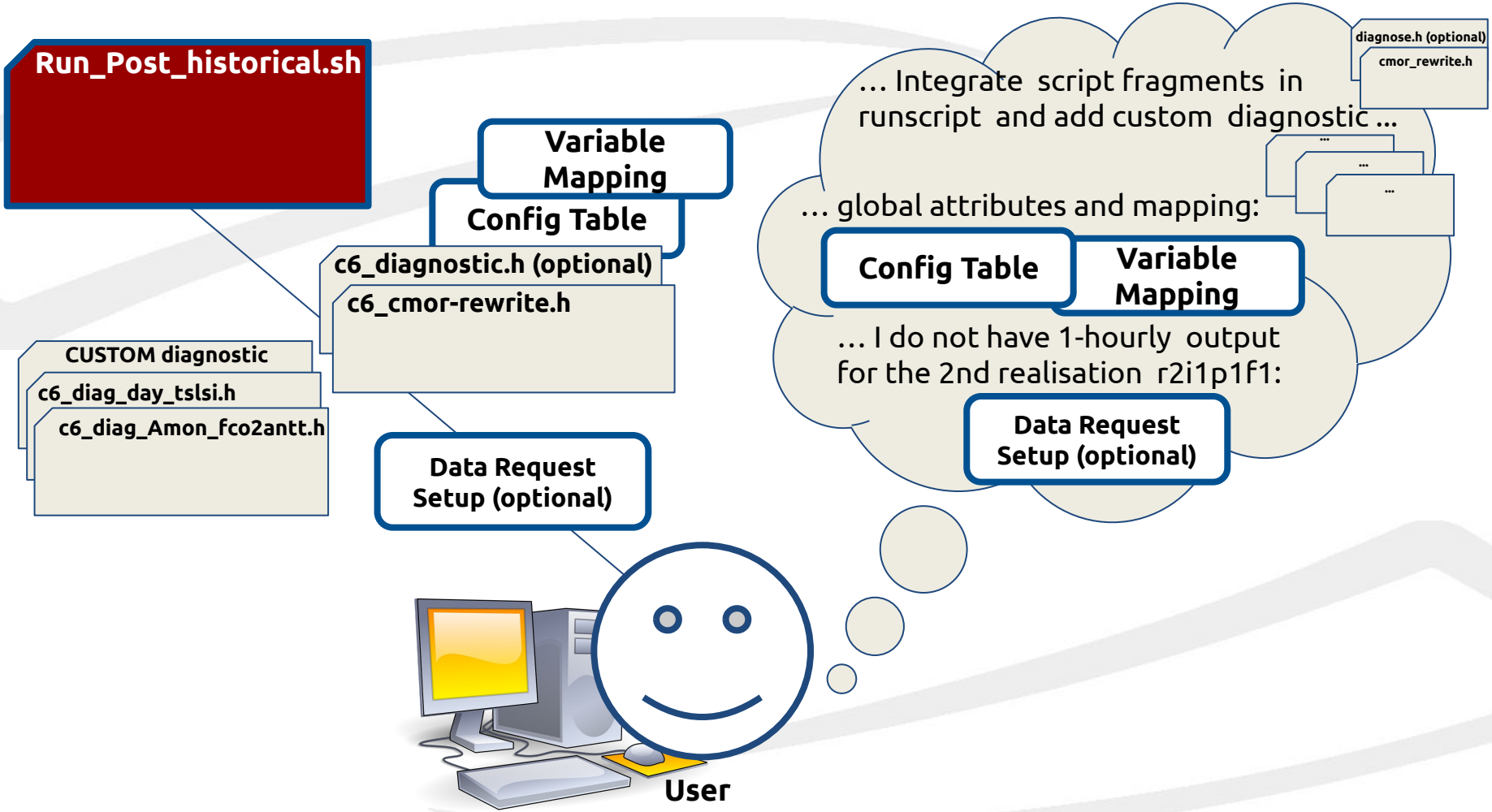
CMIP6 DataRequest WebGUI

Global attributes ...

... I run the model MPI-ESM-1-2-HR...

... I work for the MPI-M ...

User


Run_Post_historical.sh
Variable Mapping
Config Table
c6_diagnostic.h (optional)
c6_cmor-rewrite.h
CUSTOM diagnostic
c6_diag_day_tslsi.h
c6_diag_Amon_fco2antt.h
Data Request Setup (optional)
User
… Integrate script fragments in runscript and add custom diagnostic …
diagnose.h (optional)
cmor_rewrite.h
… global attributes and mapping:
Config Table
Variable Mapping
… I do not have 1-hourly output for the 2nd realisation r2i1p1f1:
Data Request Setup (optional)

Thanks for your attention!

questions to

wachsmann@dkrz.de and schupfner@dkrz.de

find support on

https://c6dreq.dkrz.de

and

https://code.mpimet.mpg.de/projects/cdo