Quality Control of CMIP5 Data

User Manual

Version: QC-0.4
Date: 2012-07-06

Dr. Heinz-Dieter Hollweg
DKRZ Data Management
Bundesstr. 45a
D-20146 Hamburg, Germany
Tel: (+49) 40 41173 546
heinz-dieter.hollweg@zmaw.de

Contents

1 General

The amount of output by coupled ocean-atmosphere general circulation models (GCM) participating the Coupled Model Intercomparison Project Phase 5 (CMIP5) will exceed anything comparable in the past. In order to make comparisons feasible, a standard was defined about the format of files and structures of archiving data. The archiving structure is defined by the CMIP5 Data Reference Syntax (DRS).[1] The conventions for climate and forecast (CF) prescribes standard names given to the variables[2] and the Climate Model Output Rewriter (CMOR)[3], a set of C-based functions, converts the output of a GCM to (netCDF) [4] files with standard meta-data.

Regional climate modelling is then done in the frame of the CORDEX project (CORDEX) driven by the CMIP5 forcing fields. CORDEX also defines a standard with requirements. [5]

A quality control (QC) of the data can no longer be carried out just by human inspection but has to be done automatically. This is the purpose of the software package QC developed at the (DKRZ). The QC checks meta-data, time axis properties, and earth system data. When very large files are sub-divided into a set of sub-temporal files, then the QC checks the consistency of time and meta-data between successive files. This is also the case for successive experiments, e.g. 'piControl' as parent and the child 'rcp45'.

Although the QC software package is designed to check climatology data (CMIP5, CORDEX), it is also capable to handle files containing data of a different structure as long as there is a certain similarity, i.e. data is given in records at time steps. The only requirement on the file is that it has to be in the netCDF format; of course without CMIP5 specific meta-data checks but only with checks of time records and data for the consistency between sub-temporal files or experiments.

This document describes how to install, configure and run the QC.

Basically, a QC run would be started by the bash script call:

```
<path>/QC-0.4/bin/qcManager -f config-file
```

[1] CMIP5 Data Reference Syntax and Controlled Vocabularies.
Taylor, K. E., V. Balaji, S. Hankin, B. Lawrence, and S. Pascoe.
http://cmip-pcmdi.llnl.gov/cmip5/docs/cmip5_data_reference_syntax.pdf

[2] http://cf-pcmdi.llnl.gov/documents/cf-standard-names

[3] http://www2-pcmdi.llnl.gov/cmor

[4] http://www.unidata.ucar.edu/software/netcdf

[5] CORDEX Archive Design. Christensen O. B., B. Gutowski, G. Nikulin.
http://cordex.dmi.dk/joomla

The life-time of the run defines a session; while 'qcManager' would launch parallel back-ground processes; each for a given variable. The back-ground processes start the core QC executable, i.e. the C++ program doing all the checks and tests.

2 Requirements

The QC does not make high demands on the IT environment. The package consists of a set of C/C++ based executables launched by bash scripts. The version of the netCDF library should be 3.6 or higher (NetCDF compressed files require NetCDFD-4). The QC runs on Linux and AIX systems.

The user must have write permissions. A QC run is feasable across different operating systems, if there is shared disk space and 'ssh' operates with public key authentication.

3 Installation

The software package is available at no charge from the (SVN) repository at the DKRZ. Please register first (send an email to legutke@dkrz.de) before you like to checkout.

svn co http://svn-mad.zmaw.de/svn/mad/Model/QualCheck/QC/branches/QC-0.4

The package layout is presented in Appendix A.

This section explains the necessary steps to make the QC run. Briefly, executables have to be compiled, a configuration file for controlling general behaviour has to be adjusted, similar a table controlling checks, tests and corresponding actions, the run of a the QC and the eventual output of resuolts.

It is important to know that almost all the files are independent on the specific project. Except the class 'QC.cpp' and the corresponding header file 'qc.h', which reflect the nature of the projects data. These however are linked easily into the QC frame.

3.1 Compilation

Make sure that the netCDF library is linked to the compiler (explicitely by options -I ... -L ..., if necessary), cd to the directory 'src' and compile the sources. Store the binaries in directory 'bin'

where the name of an excecutable is the same as that of the source with the extension '.c' and '.cpp', respectively, exchanged by '.x'; except for the main program with *qC_main.cpp –> qC.x* .

C/C++ programs must be compiled within the directory 'QC-0.4/src'. The scripts 'z_all', 'zg++' (flor linux), and 'zxlc++' (for IBM AIX) facilitate the compilation process, but unfortunately depend on ones site. Thus, adaption is necessary.

Binaries may-be disposed everywhere even outside the main QC directory. The names remain but with the extensions '.c' and '.cpp', respectively, exchanged by '.x'. In order to hold the package suitable for different project, notably CMIP5 and CORDEX, it is necessary to establish a few symbolic links prior to the compilation. E.g. if you want to compile for CORDEX, then

```
cd path/QC-0.4/include
ln -s qc_CORDEX.h qc.h
cd path/QC-0.4/src
ln -s QC_CORDEX.cpp QC.cpp
```

Eventually, the executable must get the name 'qC.x' (also, a link would be recommended when both CORDEX and CMIP5 are run from the same site).

3.2 Configuration

The QC process is generally controlled by statements from a file;
you may use 'QC-0.4/scripts/Conf/qc.conf' for a start. Options available are given in Appendix B. Some just enable a feature in a script, others are effective in the core-QC executable. At present, not up-to-date.

Checks and tests of files are controlled by a file located in 'QC-0.4/Project_table'. At present, there are the two files 'CMIP5_flags.conf' and 'CORDEX_flags.conf'. The usage is explained at the beginning of the files, followed by lines for each QC check. Each specifies the nature of the check, and allows user-defined instructions how to proceed in case a check fails: discard it, give information in the log-file state the severeness of the failure, let stop the current run or even the whole session, and notifiy by email. Additionally, the first file out of several sub-temporal files may be treated differently from the followers. Eventually, instructions may be restricted for specific variables.

Below this, the text is out of date and out of order.

3.3 Test

The directory *example* provides a test to check the installation as well as a small example of results generated by the QC. Execute the script *start_test.bash* on the command-line. This will load a configuration file from *scripts/Conf* and start the QC. If the script runs successfully in a console, then the name of the currently checked netCFD file shall appear below the executed command-line. Results of this example are written in directory *example/test*. The duration of the test should not exceed a few seconds. Notice that the purpose of *start_test.bash* is only to execute the example. For operational processing, the command

QC-path/scripts/qcManager -f configuration-file.conf [options] [&]

is used to start a run.

A failure of the test run is indicated by a missing status line below the executed *qcManager* command, missing results in *example/test*, or by an endless run (to be stopped by Cntrl C). Successive steps would help to find the reason for a failure by giving options to the call of
[../scripts/]qcManager -f qc_test.conf:

-E_CHECK_TOOLS	Display the status of executionals in the path.
-E_SHOW_CONF	Show the current configuration.
-E_SHOW_EXP	Present the path to the selected data and the corresponding experiment name.
-E_DEBUG_C &> file	Write the bash execution commands of *qcConfigurator*.
-E_DEBUG_M &> file	Write the bash execution commands of *qcManager*.
-E_DEBUG_E	Automatically active in order to log *qcExecutor* execution.
-E_NEXT -E_SHOW_CALL	Display the call of the next qC.x launch. An output indicates that the scripts processed fine. If you want to debug the qC.x file, use exactly the output.

Package Contents:

bin	Target directory for compiled executables.
bug-report.txt	Bug-fixes.
doc	Documentation.
check_list.txt	Error flags and messages issued by the QC.
user-manual	This text.

QC.html	doxygen based documentation of the programs.
example	To test and get results from a small run.
start_test.bash	Execute this to start the example run.
include	Header files.
src	C/C++ source.
qC_main.cpp	The main QC program; executable name: *qC.x*
getNC_att.cpp	Get global attributes from a netCDF file.
syncFiles.cpp	Get the next sub-temporal file to be checked.
testParentChild.cpp	Test consistency between parent and child.
testValidNC.cpp	Test validity of a netCDF file.
diskUsage.c	Show disk space usage [%]
fModTime.c	Show modification time of a file [s].
unixTime.c	Show unix time [s]
hist.cpp	Utility for converting frequency distributions.
scripts	Bash scripts launching the QC.
qcConfigurator	Parsing the configuration file.
qcExecutor	The script for launching the C++ executable.
qcManager	The script controlling the session flow.
qc.cron	Automatic restart after system shutdown.
Conf	Configuration file *qc_CMIP5.conf*.
DoW.txt	Please, notice the DISCLAIMER OF WARRANTY file.
Letters	A directory with collected e-mails describing the development of the QC package.
Project_table	Contains the CMIP5 standard table describing meta-data. This is a CSV formatted version of the original 'xlsx' table.

4 Directory tree of QC results

The configuration option *QC_DATA_ROOT* specifies the root of the sub-tree for QC results. The first component following *QC_DATA_ROOT* will be always the last path component of option *DATA_ROOT_FS*, which denotes the root of the sub-tree of the data files to be checked.

The root of the directory sub-tree for the QC results contains the sub-directories:

Project_tables	standard table and time information about experiments
data	a replication of the data sub-tree for the results.
experiment_logs	logfiles of each check of files of a given experiment. Brief logging of error flags and run-time.
log	a replication of the data sub-tree for logging execution of each data file, i.e. variable.
session_logs	logfiles of each QC session, i.e. qcManager run.
version	information about the used svn version.

The directory tree of *example* is given in the Appendix C.

5 Configuration

The flow of the QC is administered by a configuration file and options on the command-line, where the latter overrule. In fact the statements of the config-file may be used also on the command-line and vice versa. The path and name of the configuration file has to be passed via the command-line option *-f configuration-file.conf*.

A configuration file contains: key-words, assignments, and comments. Key-words are always upper-case. Assignments of multiple values may be given as comma-separated lists. A comment is anything following the '#' character to the end of the line. Empty lines, and spaces are discarded. Notice that the spelling of key-words is not checked. Unknown key-words (or typos) are ignored.

Configuration assignments follow the syntax: *key[=value[,value[,value,<newline> ...]]*

key	Enabling of a feature.
key=value	Enabling a feature and assigning a value.
key=val1,val2,. . . , valN	Passing of an array as comma-separated-list.
	An assignment may stretch over multiple lines.
SELECT/LOCK	These two key-words have a special syntax of their own:
	key [path1[, path2, ...] =] [var1[, var2, ...]]
	A path is a sub-path appended to the tree given by DATA_ROOT_FS.
	If a path is specified, then the '=' character is mandatory.
	A specified variable applies to all selected/locked paths.
	If 'path=' is omitted, then the assignments applies to variables to all paths of the DR
	Paths and variables accept Regular Expressions with the rules
	of the 'expr' command explained in 'man grep'.
	Several statements are cumulative.
	SELECT and LOCK must not be followed by the assignment character '='.

The full set of configuration option is given in Appendix B. Here, only two of them are presented, because they are mandatory.

QC_DATA_ROOT=path-to-QC-results
: Path to the root of the QC's results and logging directories. The directory sub-tree of results will always have prepended the last component of DATA_ROOT_FS.

DATA_ROOT_FS=path-to-DRS-subtree
: Path to the DRS directory tree with netCDF files to be checked.

6 Program Flow

The most direct start of the quality control is given by executing

QC-path/scripts/qcManager -f configuration-file.conf

with the mandatory two options for data targets and the path to the results. The run-time of the script *qcManager* defining a session may be performed in a console, in the background, or might

be launched by the script *qc.cron* which secures the run of *qcManager* against system shutdowns. *qcManager* scan the selected DRS directory tree for data files and builds up a comparable directory structure for the QC results. Details below are depending on the user-defined configuration.

The initial phase consists of parsing the configuration file, checking the availability of system commands, identification of all netCDF data files within the selection of variables and base directories, i.e. in general CMIP5 experiments. The script operates in cycles over the set of variables. In doing so, the sub-temporal files of each variable are checked one after another securing that the time axis given in the records of the files is strictly proceeded forward.

Any processing of a sub-temporal file by the QC issues messages to log-files and an email list. If any severe error is detected affecting the entire data set, then the QC stops. Usually, errors happen only individually for variables which are then excluded from further QC checks. Depending on the configuration, errors in the meta data and non-time data will throw warnings without exclusion from further checks. However, errors in terms of time and date will stop the processing of the sub-temporal data set.

The purpose of *qcManager* is just to handle the scheduling of the QC. It launches the script *qcExecutor* for each sub-temporal data file to be checked, which then controls the execution of the C++ program qC.x that accomplishes the real checks. *qcExecutor* runs as an asynchronous process in the background, several instances may run (the number is limited by the configuration setting), and even instances may run on different platforms if these share the file system. Each *qcExecutor* communicates with *qcManager* by emitting bash signals.

7 Check Outline

The QC is primarily designed to check data of CMIP5 experiments and meta-data which are described at http://cmip-pcmdi.llnl.gov. But, the software package allows that modellers add variables to their set of data that are not defined by the CMIP5 standard table or that the package is used to check data that are of non-CMIP5 origin at all. Therefore, QC builds up a so-called *Project Table* from properties found in the checked files. This table is later consulted to ensure consistency between sub-temporal data files and across experiments.

However in the first instance, QC compares meta data of a variable against the standard table when it is checked the very first time. The checks are grouped for CMIP5 data sets as:

Filename syntax: DRS encoding of the name is checked and each component of the filename is compared to the global attributes in the file. The first item is the acronym of the variable and when not found in the table, then a warning is thrown. Checks follow for the items MIP table name, mode, experiment, and ensemble member (MIP table: defined by 'table_id' in http://cmip-pcmdi.llnl.gov/cmip5/docs/CMIP5_output_metadata_requirements.pdf). The time stamp of the name of sub-temporal files is checked against the dates of the first and last record in the file.

Variable: The acronym of the file is identical to the *output variable name* in the table (italic font denotes columns of the table) and the table entry is found by matching MIP table names. Thus, these two entries cannot be checked. Checked are:

- *long name*
- *units*: more precisely the *unformatted units*. Units must match exactly.
- *standard name*
- *cell-methods*: discarding items if parenthesis in the table and looking for a match of each remaining item in the file's attribute.
- *cell-measures*: see cell-methods.
- *type*: 'real' of the table is a synonymical to 'float' in the file.
- *CMOR dimensions*
- *realm*.

Dimensions of a Variable: The dimension name of the MIP tables and the corresponding one in the file are usually different. But they are linked by the 'dims' sheet of the standard table, where the *output dimension name* denotes the name used in the file. Checks are then accomplished for:

- *standard name*.
- *long name*.
- *axis*.
- *units*: for 'time' the kewords 'days since' must be given in the file units.
- *bounds?*: (the '?' belongs to the heading).
- *type*.

- *value*.
- *bounds values*.
- *requested*.
- *bounds_requested*.

The **size** of a dimension is only checked, when the tables specifies a fixed number of required values.
The **values** of a dimension are checked by the checksum (Fletcher, J. G., An Arithmetic Checksum for Serial Transmissions, IEEE Trans. on Comm., Vol. COM-30, pp 247-252, 1982). The checksum is stored in the Project Table in the 'Auxiliary' section.
Only the **17 mandatory levels** of dimension 'plevs' are checked.

Auxiliary Variables: These are usually dimensions represented by a variable. Or in some instances a variable without a corresponding dimension in the file, but with a dimension of size equal 1 in the table. Only if this type of variable is not time-dependent and is not specified as 'CMOR fixed variable', then basic properties are stored in the Project Table.

- *variable name* of the auxiliary.
- *dimension names*.
- *checksum* of the values.

Project Table: An entry of this table is a 'finger print' of each variable. It consists of sections for properties of the variable, the corresponding dimensions, and the auxiliaries. The MIP table and a possible sub-table (10 significant characters) are specified, too. This finger print must match for all sub-temporal files of a given experiment. And if a parent experiment is given, then also across the experiments. This is also required for non-CMIP5 data sets.

8 Time Data Checks

In a regular mode, i.e. there is a parent and a constant time-step, the following items are subject to checks:

- *negative time step*
- *identical time step*
- *missing time step*
- *negative/zero time-bounds range*
- *overlapping time bounds ranges*: between records and across files.
- *gap between adjacent time bounds ranges*

Whether the mode is a regular one is prescribed by the time schedule information about the various experiments in the standard table. A few experiments have no regular time-step, others (e.g. RCP) have a gap of decades, etc. . The time schedule for experiments was extracted from the standard table and is stored in *QC/Project_tables/time_table.csv*.

9 Earth System Data Checks

The dimensionality of earth system data is not explicitly taken into account. Each of the following operations is accomplished over an entire record. The time series of values are written to a qc-result file; a netCDF file as atomic data set over the entire experiment for each variable.

Estimated for each record are the **minimum**, **maximum**, **average**, **standard deviation**, and the **number of grid-cells with filling value**. The value of the average depends on the configuration setting; the average is weighted by triangulation or just by the frequency of cells assumed to be of equal area.

The following flags could indicate detected errors or warnings.

- **entire record with filling value**
- **undefined standard deviation**
- **entire record with constant value**
- **suspect minimum**:

$$\mid \textit{ave\_min} - \min(v_{r=N}) \mid > \textit{magnitude(ave\_min)} \times 10^5$$

  with current record number N and grid-cell value v and the average

$$\textit{ave\_min} = \frac{1}{N-1} \sum_{r=1}^{N-1} \min(v(r))$$

  It is expected that the minimum is negative; otherwise the test for the maximum would fail (and vice versa). The value 10^5 is chosen arbitrarily.

- **suspect maximum**

$$\mid \textit{ave\_max} - \max(v_{r=N}) \mid < \textit{magnitude(ave\_max)} \times 10^5$$

- **suspecting a replicated record**: This is given when minimum, maximum, average, and the standard deviation are equal between two records (from the first record of the experiment until the current one under investigation; equality in the sense that the absolute value of the deviation of the two is smaller than the magnitude times 10^{-6}). Records entirely of filling values are excluded.

10 Appendix A

QC Software Package (only the basic files):

Package Contents:

bin	Target directory for compiled executables.
bug-report.txt	Bug-fixes.
doc	Documentation.
user-manual	This text.
QC.html	doxygen based documentation of the programs.
example	To test and get results from a small run.
start_test.bash	Execute this to start the example run.
include	C/C++ Header files.
src	C/C++ source.
qC_main.cpp	The main QC program; executable name: *qC.x*
getNC_att.cpp	Get global attributes from a netCDF file.
syncFiles.cpp	Get the next sub-temporal file to be checked.
testParentChild.cpp	Test consistency between parent and child.
testValidNC.cpp	Test validity of a netCDF file.
diskUsage.c	Show disk space usage [%]
fModTime.c	Show modification time of a file [s].
unixTime.c	Show unix time [s]
hist.cpp	Utility for converting frequency distributions.
scripts	Bash scripts launching the QC.
qcConfigurator	Parsing the configuration file.
qcExecutor	The script for launching the C++ executable.
qcManager	The script controlling the session flow.
qc.cron	Automatic restart after system shutdown.
Conf	Configuration file *qc_CMIP5.conf*.
DoW.txt	Please, notice the DISCLAIMER OF WARRANTY file.
Letters	A directory with collected e-mails describing the development of the QC package.
Project_table	Contains the various tables for configuration.
CMIP5_flags.conf	Check controlling.

CMIP5_standard_output.csv Standard specification.
CMIP5_time_table.csv Time schedule specifications.
CORDEX_archive_design_1.0.pdf Standard specification.
CORDEX_flags.conf Check controlling.
CORDEX_requirements.csv The requirements mentioned in 'CORDEX Archive Design'.
Note: CSV is a formatted version of an originally 'xlsx' table.

11 Appendix B

(Note: items in [] denote a default)

QC_DATA_ROOT=path-to-QC-results No default, mandatory

Path to the root of the QC's results and logging directories. The directory sub-tree of results will always have prepended the last component of DATA_ROOT_FS.

DATA_ROOT_FS=path-to-data No default, mandatory

Path to the DRS directory tree with netCDF files to be checked.

APPLY_MAXIMUM_DATE_RANGE=... [disabled]

This option controls whether the second time-stamp in the filename expresses the sharp end (deadline) of the period or whether the period in the stamp has to be extended to the end, e.g. 2000-03 <=> 2000-03-31 for extended. The difference is explained best by examples.
Sharp: 2000 - 2001 <=> 2000-01-01T00:00:00 - 2001-01-01T00:00:00
Extended: 2000 - 2001 <=> 2000-01-01T00:00:00 - 2001-12-31T24:00:00
If there is a mix for the components of the stamps, then use this syntax: 'yx-mx-dx-Hx-Mx-Sx' with x=s or x=e for sharp and extended, respectively. Omission of any component sets individually x=s. E.g. APPLY_MAXIMUM_DATE_RANGE=ye-ms <=> ...=ye-ms-ds-Hs-Ms-Ss. Sharp is applied as default.

ARITHMETIC_MEAN [disabled]

By default, calculation of cell averages over a geographical grid is performed by weighting with the corresponding grid-cell area. In particular for ocean basin cross-section, this is very time consuming due to the irregular shapes of the basins taken into account by varying filling values (also for regular grids with varying effective areas, e.g. T of snow). If the precise value of the average is less important; this option reduces calculation time.

CHECK_MODE=time [data]

The QC checks meta data, time data and data (of variables), where for the latter two the records are inspected. Modes can be enabled by key-words: meta, time, data. Note: data includes meta and time; time includes meta.

CHECKSUM=executable_name [disabled]

Calculate checksum of every file that passes successfully the QC. If md5 or sha1 are assigned,

then a separate text file is generated where the checksum is stored with appended extension md5 or sha1, respectively, to the filename, i.e. filename.nc.md5. Else, a named script/program is executed. Such a script/program must be available in the user's search path for commands or the path has to be absolute. The function is not part of the QC package. The calling qcExecutor script provides the following parameters:

1) filename
2) path to the root of the CMIP5 directory tree
3) path to the data file in the CMIP5 directory tree
4) path to the session-log file
5) name of the experiment as defined by DRS_TREE_INDEX

If the script/program outputs two elements to standard output (suppose: filename and a kind-of-checksum), then these are written to a table in the project path named according to the experiment-log-name prefixed by 'cs_'. Select the checksum algorithm (md5 | sha1 | function/program)

CHECKSUM_DYNAMIC_TIME_STAMP [enabled]

If the range denoted by time-stamps of a file grows during the process of file creation and additionally option CHECKSUM is enabled, then the detection of a completed sub-temporal data set is not sufficiently given by comparing the time-stamp period with the time values in the file. This option will identify a sub-temporal data set as complete, when there is another file available with an adjacent time-stamp.

CHECKSUM_FINAL [disabled]

Checksums are usually calculated, if the QC detects no error AND a range of dates is appended to the filename that fits with defined uncertainty to the time range in the netCDF file. This option here applies to files without time-stamps. A checksum is calculated if the QC detects no further modification of the file and there is no error-file in the path. Note: this option should only be switched on when the model has reached the final end of an experiment.

CLEAR[=key] [disabled]

Remove all former results corresponding to the paths and variables selected before re-doing the QC. However, corresponding logfiles are not affected. If key-word 'only' is assigned, i.e. CLEAR=only, then no QC will take place. If key-word 'err' is assigned, then only those will

be cleared that are blocked by a file qc_error_<filename>.txt . If key-word='resume', then the qc_error_<filename>.txt file will be removed before processes will be continued. Behaviour for symbolic links: these are removed by default, but not the targets. If targets should be removed too, then use the key-word 'follow_links' (or 'follow', or 'fl'). Application of more than a single option has to be specified as comma-separated-list where the order doesn't matter.

DEREFERENCE_SYM_LINKS [disabled]

The original directory tree structure of the source data is preserved, i.e. a symbolic link in the tree will make a corresponding symbolic link in the directory tree of the QC results, too. Enable this option, if symbolic links are to be dereferenced, i.e. genuine data are to replace the symbolic links in the QC result tree.

DISABLE_FILENAME_CHECK [enabled]

If the project_id in the data file header specifies 'CMIP5', then a check is performed between the components of the filename and all corresponding attributes specified in the file. If the project is CMIP5, but you know that the filenames do not accord to the DRS filename syntax, then you may disabled the check.

DISCARD_LOG_OF_GOOD_EXEC [disabled]

Facing trillions of files to be checked, the logging of executions statements of successful checks would allocate a lot of disk space. With this option, the QC result log-tree would only contain logfiles for abnormally exited qC.x runs. Note: this does not apply to experiment and session logfiles:

DISTRIBUTED_FS [disabled]

If distributed computing is also involved with distributed file system, then enable this option. Bi-directional communication and data transfer is done by ssh and scp, respectively. IMPORTANT NOTE: this is not installed and not projected for CMIP5. However, the QC for CMIP5 may have distributed computing with a shared file system.

DRS_TREE_INDEX=... ['unknownExp']

Note: this feature is not essential for the QC. The purpose is just to tag a larger set of various checks with a name that contains a certain volume of netCDF files. For instance,
'HadCM3_historical_r3i2p1_v1' would contain all variables in the named frame. This tag names the experiment-logfiles that collect all messages. The tag is created automatically at each check of any variable. Thus, one could in principle run the QC over the entire CMIP5 set

and the messages are sorted in their respective tagged experiment-logfile. A tag can be constructed by any path component according to the Data Reference Syntax (DRS):
<activity>, <product>, <institute>, <model>, <Experiment>, <frequency>, <realm>,
<Variable_name>, <ensemble_member>, <version>, and <MIP_table>
The user has different ways to determine how a tag would look like:

I) rename the default (which is 'unknownExp') by a string that must have more than a single leading letter.

II) By indexing the position of the path component, i.e. the string between two adjacent slashes (/string/). The index number runs backwards, e.g. in '/one/two/three' 'three' would get 1 and 'one' gets the index 3. The comma-separated sequence '3,1' would be applied from left to right with the result of a tag 'one_three'. So, you have to count the components of the path to your data from the deepest level.

III) Use the first character of the DRS syntax components as they are given above, where case is significant (notice uppercases in Experiment, Variable, and MIP_table). Append the respective index number to each letter. The comma-separated list is independent on the order.

If any index number is higher than the number of components of the path to the data files, then the default tag will be used.
IMPORTANT: This does not select any path for checking.
DRS_TREE_INDEX=allCMIP5 example for I)
DRS_TREE_INDEX=4,1,8 example for II)
DRS_TREE_INDEX=i9,m8,E7,e3,v2 for the ESGF data node directory structure.

EMAIL_TO=some.account@site.az [disabled]
In case of error: send e-mails to this comma-separated-list. Submission of e-mails might be toggled on/off during run-time by sending the signal 'kill -USR2 pid', where pid indicates the process ID of the qcManager process. Please, do not confuse with USR1.

EMAIL_END_OF_SESSION [disabled]
Notify the finish of a session by e-mail.

EXCLUDE_VARIABLE=variable-list [disabled]
Exclude specific variables globally from a check.

FILENAME_PATTERN_VAR =RegExp [disabled] Variable name and sub-table names are found in CMIP5 DRS compliant filenames by default. However for non-compliant filenames, the two can be supplied by pattern matching with regular expressions. IMPORTANT: to check non-compliant, non-time dependent variables, so-called fixed fields, specification of FILENAME_PATTERN_VAR is the only way. The default for a non-compliant sub-table is 'Any'. Here is an example for a file 'yxc.asdf.pr_A1_1.nc' with variable 'pr: `.\*\.\(.\*\)_.\*_.\*.nc`

FILENAME_PATTERN_SUBTABLE =RegExp [disabled]
As above. The example for sub-table A1: `'.*\_\(.*\)\_.*.nc'`

FINE_META_DATA_CHECK_REPORT[=all] [disabled]
Detected errors/warnings due to meta data checks (variables and dimensions) are merged by default on a single line in the logfiles. To have these separated and more detailed, enable this option. If 'all' is assigned to the key-word, then all generated error/warning issues are also written to the qc_error_<filename>.txt and qc_warning_<filename>.txt.

FLOW_TRACE [disabled]
Enable an approximate flow-trace analysis of the main loop of qcManager

FREQ_DIST [disabled]
Calculate the frequency distribution of the entire domain Note: generally, you will need post-processing with 'hist.x' to assemble the final frequency distribution, except you select specific debug options. Rules for selecting class widths and starting point (e.g. centring): Firstly, a fd-build-file of the same variable at destination has precedence. Second choice is a fd-build-file or fd-prop-file (the header of a build-file) in a specified path. Thirdly, explicit statement of properties.

FD_TIME_PART=int[U] [the entire experiment]
Partitioning the total time span of an experiment into smaller ranges. FreqDists will be calculated separately for each window. Note: the residual time interval may be shorter. Note: if no unit-designator U is appended (y, mo, d, h, mi, s), then the unit from the time variable is used.

FD_PROPERTY_PATH=... [disabled]
By default, frequency distributions adjust automatically the bin (class) width while processing the first time window (or the total time span), which is then used in subsequent time windows for the entire experiment. Specifying this path enables to retain the properties of the same

variable (if found) of another experiment, e.g. control run. Such a file must have extension .build or .prop

FD_EXPLICIT_PROPS=W/I [disabled]

Explicit properties (two floating numbers separated by '/'): class-Width () / Init value (number) e.g. 1/0 will centre a bin (class) of the frequency distribution around zero. Note: The alignment of the bin borders is done automatically. It is not possible, do have a class boundary exactly at say 273.15 with class-width of 1. This option is only reasonable for selecting a particular common type of variable in a separate run, e.g. temperature. See priority rules.

FD_PLAIN [disabled]

Output of the Freq Dist as ready-to-use. By default, the output will be in a format to be re-read later in order to resume a previous session (the output file gets extension '.build'). File extension in plain mode: '.hist'. Attention: this starts and completes a FD calculation. Makes little sense for multi-sessions. Purpose: debugging

FD_BARS [disabled]

Like FREQ_DIST_PLAIN, but output of the Freq Dist shows complete shapes of bars. Purpose: debugging

GROUP_NAME=... [disabled]

Set group name. This is necessary, if the QC is also operated by users who are not in the default group of the user who initially checked out from the svn repository. This will automatically also set the SGID-bit, i.e. grant permissions to all group members. The current setting of user permissions of each file is duplicated to the respective group permissions. Note: this can only be done by the owner of the QC directory.

HARD_SLEEP_PERIOD=int [10 s]

In order to enable trapping signals, long sleeping period are subdivided into smaller intervals of consecutive sleep commands.

HLEVD [disabled]

The Hans-Luthardt-Extreme-Value-Detection (see code in qcExecutor). The HL extreme value is expressed as deviation range normalised to the units of standard deviation: (max-min)/stdDev, i.e. is dimensionless for each grid-cell. If a threshold is exceeded, then by default, a notification is written to the SESSION_LOG. Makes not really sense for applying to 3D. Derived by and therefore requires CDO.

HLEVD_DELETE_FILE [disabled]

Delete netCDF file with HLEV in any case; only notification takes place.

HLEVD_DETREND [disabled]

To detrend a time series takes some time. If a trend of a variable does not boost exponentially, detrending should not affect this kind of extreme value detection (which is anyway rather vague).

HLEVD_KEEP_THRESHOLD_FILE [disabled]

Keep netCDF file with HLEV if threshold was exceeded.

HLEVD_KEEP_THRESHOLD_FILE [disabled]

Keep netCDF file with HLEV exceeding threshold.

HLEVD_THRESHOLD=int [1]

Threshold: (max-min)/stdDev > threshold

IGNORE_LOCK_FILES [disabled]

An error in any record of a given variable activates a specific management for this particular variable. The file with the error is completed, but any file of the variable with a following temporal subset will not even be touched. This option ignores any blocking of following temporal subsets. Note: In a situation that a blocking file was found, but no qc_<filename>.nc file, then the blocking holds anyway.

IGNORE_PROJECT_TABLE_ERR [disabled]

Usually, detecting an error in the netCDF layout in comparison to the project table lets stop the QC and an error-file is generated blocking any further analysis of the respective variable. This option converts error-files into warnings and thus, cancel the blocking. I hope, you have read the disclaimer-of-warranty! Note: Real run-time errors shall not be ignored. Note: Failed checks against a standard table issue always warnings.

LOCK [path1[,...=]][[var1],...] [disabled]

Locking of variables (LOCK takes precedence over SELECT). See SELECT for a description.

NEXT_RECORDS=int [disabled]

Only progress by the given number of records at each check. Purpose: probably debugging

NICE=int [disabled]

Apply linux 'nice' command to executables on guest machines (see 'man nice').

NIGHT_SHIFT [disabled]

Suspend processes between 8 - 19 o'clock on guest hosts and allow only a single process on the qcManager host.

NON_REGULAR_TIME_STEP [disabled]

This disables any check related to a constant time step within a file, across files, or across experiments. In fact, the only property of time to be checked is that time marches on and that time bounds don't overlap.

NUM_EXEC_THREADS=int1[,. . . , intN] [1]

List of number(s) of simultaneous execution processes per host. Note: there is always a single qcManager process. Special: If a single number is specified, this is assigned to all hosts in the QC_EXEC_HOSTS list. Fine tuning: each positional number in the list corresponds to a position in the QC_EXEC_HOSTS list. If less positions are given than in QC_EXEC_HOSTS, then the last position is assigned to the omitted positions.

PARENT_CHILD_CHECK_ONLY [disabled]

The QC checks for each first sub-temporal file of each variable the continuation from a corresponding file of a parent experiment before a regular check of the child. This option disables any regular QC. Thus, only the parent-child crossing is checked.

PARENT_EXP_ID=string [from attribute 'parent_experiment_id' in the child]

Usually, the QC checks for continuity between parent and derived experiments by using the global attribute 'parent_experiment_id' from the netCDF header. However, knowing the parent experiment is not sufficient for a child to infer a unique path to its ancestor. The algorithm applies the following rules:

1) PARENT_EXP_ID=none skips any parent-child relation test.

2) From global attributes of the netCDF file: parent_experiment_id and parent_experiment_rip.

3a) If 2) fails, then PARENT_EXP_ID and/or PARENT_ENSEMBLE_MEMBER.

3b) PARENT_ENSEMBLE_MEMBER not specified: same as of the child.

4) PARENT_VERSION as specified; else: same as of the child.

5) Parent and child must be located in the same DRS directory tree with similar setting, but experiment_id and version.
6) The last sub-temporal file of the parent must be available. If this fails, then the QC will throw an error flag.
7) The QC checks the validity of the path to the parent file. If invalid, then a notification is given as usual. It is due to the user that the rules apply.

PARENT_ENSEMBLE_MEMBER= [disabled]
This parameter should be defined in the global attributes of the netCDF file as 'parent_experiment_rip'. This is looked for as the default. If absent, then the ensemble member of the current experiment is tried. Set option, if nothing fits (e.g. r1i1p1).

PARENT_EXP_RIP [checked]
The attribute parent_experiment_rip is always checked by default. If the attribute is missing throughout the entire experiment, the check is skipped by disabling. A check is also skipped, if the option is a assigned the string 'none'.

PARENT_VERSION=string [same as of the child]

PROC_POOL=directory-name [automatically]
A temporary directory for locking files and communication between qcManager and qcExecutor. Usually, this will be created automatically for each experiment and released afterwards. If the path is relative, then it refers to QC_SRC.

QC=on/off [on]
Switch on a Quality Control run. It is possible to switch it off, for instance to calculate only frequency distributions. (not a good example, because both can be run simultaneously).

QC_BIN=host1:bin,bin,host2:bin2 [bin]
Executables reside in QC/bin by default. For the purpose of running the QC from another computer with the same file system, there may be a different directory for the binaries specified. If the path is relative, this bin-dir has to be a sub-dir in the path to QC. If directories are prefixed by the hostname(s) of computer(s) separated by ':' from the directory, then multiple directories may be specified as comma-separated-list. The name(s) of the computer(s) must be available in the list of hosts QC_EXEC_HOSTS. A name without a prefix is dedicated for the host on which qcManager is running (but could also have a prefix). You have to compile all executables into all specified bin-directories.

QC_EXEC_HOSTS=host1[,...,hostN] [$HOSTNAME where qcManager runs]
A list of machines executing asynchronous jobs.

QC_MANAGER_HOST=your-hostname [$HOSTNAME]
The machine where a session is started; usually the user's pc.

QC_PROJECT=name [CMIP5]
The name of the game: attribute *project_id*
Disabling suppresses related meta-data checks.

QC_VERSION [disabled]
Write the svn version-numbers of each package file to a file for each session. Located in the QC_DATA_ROOT/.../session_logs.
Name: qc_version_yyyy-mm-dd_hh:mm:ss.log
Note: if there is no svn, then names and modification dates are written.

REATTEMPT_LIMIT=int [5]
If a process cannot be run, because e.g. the data base is not available or a server is absent, then, retry after a sleep of the specified duration given in seconds.

REDO=string [disabled]
Recalculate a quality check for all selected variables. Assigned key-word 'save' acts like option CLEAR, but with the difference that former results are saved in a sub-dir REDONE.num, where num is an increasing cycle number. If a range is in terms of dates, then the two dates are separated by '-'; times have to be separated by '/'. For this particular feature, no former results are saved, because the qc_<filename>.nc will change. If the second parameter of a time range is missing (however, '/' or '-' is still required to solve a potential ambiguity), then redo from the specified value to the end of the qc_<filename>.nc.
REDO=save: save before redoing all selected variables.
REDO=t0/t1: for a time range.
REDO=d0-d1: for the CMIP5 formatted range of dates.

SELECT [path1[,...=]][[var1],...] [disabled]
Selection of paths and variables (RegExp of the 'expr' command, i.e. full specification from the beginning of the word). Omission for paths selects the selected vars in all paths of the CMIP5 DRS directory tree. Omission of variables selects all variables in the specified path(s). Notice the '=' sign at the end of (the comma-sep list for) path(s).

SELECT path=var [disabled]
SELECT pr_* all variables named pr_What-Ever-You-Like
SELECT .*/6hr,.*/mon/atmos/=v1 example for a path selection
SELECT := .*/v?[[:digit:]]3= example for all DRS versions

SLEEP_PERIOD=int [300 s]
Sleep period if waiting for a server or anything else. Note: particular ordinary processes have a sleep time of their own to distinguish them in the output of 'ps'.

STOP_AT_TIME_ERROR [disabled]
By default, errors in the time variable will not abort the QC immediately, but the QC continues to check until the end of the current sub-temporal data file. This option stops execution right in front of the corrupted record.

SYNC_FILE_AMBIGUITY_CHECK [disabled]
By default, the QC examines begin and end dates within sub-temporal files and finds the right one for proceeding (or starting) a check by synchronisation. The purpose of this option: Check that no older sub-temporal data files are in the path that would in fact fit properly into the sequence of begin and end dates of the files. This option checks also modification times of the files, thus keep this option disabled, if the sub-temporal files could be copied in arbitrary sequence. Enabling is useful if the QC goes along with the process of creating data files and existence of remnants from previous tries can not be excluded.

TABLE_RELAXED_DIM_CONSTRAINT [disabled]
Usually, the table prescribes a direct dimensionality of the grid-layout. However, a wider scope is often available in models (e.g. parameterised dimensions in ocean models depending on multiple arbitrary dimensions). This option allows for such a wider scope and a dimension is only checked if it is defined with identical layout within both table and file.

TABLE_RELAXED_VAR_CONSTRAINT [disabled]
Raise only warnings for errors in the attribute of a variable.

TABLE_VARIABLE_CONSTRAINT [disabled]
Variables must be declared in the standard table. The default behaviour is described below. If there is any variable in the model run that is not part of CMIP5, then disable this.

TABLE_PROJECT=... [rule 2)]
Standard and project tables are used to check consistency and continuity of parted atomic-data

sets and experiments. If there is no predefined standard table, then a project table is created with the properties found in the file at the first check. Once a project table contains a specific entry for a variable and realm, then later checks are against this entry. TABLE_PATH is identical to QC_DATA_ROOT/... by default. Initially, the standard table is copied from QC_SRC/Project_tables. The default path may be changed by a leading path in front of the name of the standard table taking precedence over the default behaviour. TABLE_PATH=... [$QC_SRC/Project_tables]

Rules for naming the project table:

1) precedence of a fully qualified project name 'anything'.csv
2) incomplete project table name and standard table name:
   TABLE_PROJECT="incompleteName"_standardTable
3) incomplete project name and no standard table:
   TABLE_PROJECT="incompleteName_table.csv", however, "_table" substring omitted if "table" is already available.
4) no project table, but standard table:
   TABLE_PROJECT="project_"standardTable
5) no project table and no standard table:
   TABLE_PROJECT="project_table.csv"

TABLE_STANDARD=standard_output_12Jan2011.csv [none]

Converted from the standard table available at the denoted date. Format: comma-separated-values

TIME_LIMIT=date [disabled]

Stop the QC at a specified time. Use ISO-8601 format (yyyy-mm-ddThh:mm:ss). Truncations are accepted (1955-06 would process the records before 1955-06-01 00:00:00). If the parameter gets a leading 't' or 'T' character, then a time value (a pure number) is accepted that must correspond to the reference date in the data file.

WAIT_FOR_MISSING_DATA [disabled]

A gap between the last record of a file and the first record of the next file is treated as an error by default and a message is written to a qc_error_<filename>.txt . This option enables to wait for the closing of the gap. Purpose: sub-temporal data files are written or uploaded in arbitrary

order. Note: a real gap between two files will never be detected and the QC waits forever (however, the process will finish normally, because nothing has changed).

WORK_AT_LOW_LOAD [disabled]

Start the analysis on a guest machine, if the average load from the uptime command is < 1.5 for 5 minutes and < 1 for 15 min. At present, this option is disabled permanently.

12 Appendix C

The directory sub-tree of the test in directory *example*:

QC_DATA_ROOT=path/example/test
DATA_ROOT_FS=path/data/CMIP5

==> path/example/test/CMIP5/

```
config
     qc_test_2011-03-29_14:11:28.conf
data/output/inst/model/rcp45/mon/atmos/tas/r1/
     qc_tas_Amon_model_rcp45_r1.nc
     qc_warning_tas_Amon_model_rcp45_r1_195511-195609.txt
     qc_warning_tas_Amon_model_rcp45_r1_19550101-195511.txt
     qc_warning_tas_Amon_model_rcp45_r1.txt
data/output/inst/model/rcp45/mon/atmos/pr/r1/
     qc_warning_pr_Amon_model_rcp45_r1.txt
     qc_error_pr_Amon_model_rcp45_r1.txt
     qc_pr_Amon_model_rcp45_r1.nc
experiment_logs
    unknownExp.log
    unknownExp.time
log/output/inst/model/rcp45/mon/atmos/tas/r1
    tas_Amon_model_rcp45_r1_195511-195609_2011-03-29_14:11:53.log
    tas_Amon_model_rcp45_r1_2011-03-29_14:11:56.log
    tas_Amon_model_rcp45_r1_19550101-195511_2011-03-29_14:11:44.log
log/output/inst/model/rcp45/mon/atmos/pr/r1/
    pr_Amon_model_rcp45_r1_2011-03-29_14:11:49.log
Project_tables
    standard_output_12Jan2011.csv
    pt_standard_output_12Jan2011.csv
session_logs
        qc_test_2011-03-29_14:11:28.log
version
    version_2011-03-29_14:11:28.txt
```