QC2 Exception Codes (preliminary version)

Fatals: immediate action neccessary,
errors: data unacceptable,
warnings: data possibly not ok,
informatory: fine - just for info,
unclear: title lines or open issues.

Thematic exception groups (see first column):
ACCEss errors, **GENE**ral checks, **META**data and file name checks,
TABL: inconsistencies in comparison to meta data tables,
TIMEaxis checks, **VARI**ables' checks.
OBSOlete messages (not used for CMIP5 project).

The following flags F<n> refer to general checks,
mainly on the time axis.

key group	description	comment
F-1 GENE	-- Not checked	
F0 GENE	-- No error found	
F1 TIME	testTimeStep() ^0, ^5 Error: negative time step	
F2 TIME	testTimeStep() ^0, ^5 Error: missing time step	This, of course, is no error if the QC is run over several time slices with intentional gaps in between. You may want to set NON_REGULAR_TIME_STEP in the setup file to check only for positive increments (of perhaps different sizes).
F4 TIME	testTimeStep() ^0, ^5 Error: identical time step	
F8 TIME	testCalendarTimeBounds() Error: negative/zero time bounds range	
F16 TIME	testCalendarTimeBounds() ^0 Error: overlapping time bounds ranges	
F32 TIME	testCalendarTimeBounds() ^0 Warning: gap between time bounds ranges	This warning refers to the bounds CF defines for every time step within a file. Rightbound(n) should be = Leftbound(n+1).
F100 VARI	testData() Warning: found a record entirely with filling value	
F200 VARI	testData() Warning: found a record entirely with constant value	
F400 VARI	testData() Warning: suspect minimum Note: no table comparison; inferred from the data	
F800 VARI	testData() Warning: suspect maximum Note: no table comparison; inferred from the data	
F1600 VARI	testData() Warning: undefined standard deviation	
F3200 VARI	testData() Warning: suspecting a replicated record Note: a record of min, max, ave, and stdDev is identical to a previous one. Note: fields of constant or filling value excluded.	For most variables this should not happen. Check if 2 consecutive steps may have identical values. The comparison of the values is: abs(diff) < 10**-7 .

Some of the following exception codes interrupt the check process.

Ordered by keys. **(This list ordered by groups is here).**

key group	description	comment
1 OBSO	from qc_main.cpp: -- Internal errors indicating flaws in the scripts or qc_main.cpp.	All the errors of the 1_ family should not happen unless scripts or qc_main.cpp was changed. They may happen in test or debugging situations.
1_1 OBSO	from qc_main.cpp: getFilename() ^2 Invalid name.	...in script qcExecutor (see comment of exception 1).
1_2 OBSO	from qc_main.cpp: insertPointedObj() ^2 String parsing: definition of object ... must not have references.	...in the c++ code (see comment of exception 1).
1_3 OBSO	from qc_main.cpp: getopt() ^2 Undefined option ...	...in the c++ code (see comment of exception 1).
1_4 OBSO	from qc_main.cpp: setObjLinks() ^2 ... must have a single reference.	...in the c++ code (see comment of exception 1).
2 OBSO	from qc_main.cpp: entry() ^2 No QC object linked to InFile object.	...in the c++ code (see comment of exception 1).
5 OBSO	from Base.cpp: setFillingValue() Undefined explicitly provided variable name. Note: not related to CMIP5.	
6 OBSO	from Base.cpp: setVarPropsNoOperation() No rules to link alias to any variable in operation. Note: not related to CMIP5.	
7 OBSO	from Base.cpp: setVarPropsNoOperation() No rules for finding a variable name. Note: not related to CMIP5.	
8 OBSO	from Base.cpp: setVarPropsNoOperation() No rules for finding a variable name. Note: not related to CMIP5.	...in the c++ code (see comment of exception 1).
9 OBSO	from Base.cpp: getVarname(std::string &s, std::vector<std::string> &) ^2 Invalid assignment statement of variable names. Note: not related to CMIP5.	
10 ACCE	from InFile.cpp: init() ^2 Could not open netCDF file. Note: non-operational processing only.	NetCDF output file for (QC results) could not be opened. This is checked before, too - so there normally should be no access errors here.
12 VARI	from InFile.cpp: checkVarType() The type of the variable is not float. Note: CMIP5 requires always float.	The checked variables are expected to be float! Double variables are not checked but their error messages can be suppressed in setup file. Unchecked variables cannot be part of a DOI (persistent Document Object Identifier).
14 TIME	from InFile.cpp: setGeoParam() Dimension rank of the field is higher than 3D. Note: the dimension of time is excluded.	Three space dimensions are allowed as maximum.
15 VARI	from InFile.cpp: setGeoParam() Vertices or bounds of unknown format.	
16 VARI	from InFile.cpp: scanNcVars() ^2 Explicitly provided variable name not found in the netCDF file. Note: non-operational processing only.	The expected variable was not found in the NetCDF header.
18 VARI	from InFile.cpp: xtractNum() Dimension rank of the field is higher than 3D. Note: the dimension of time is excluded.	2D (lat/lon) and 3D (lat/lon/alt) data are expected.
19 GENE	from InFile.cpp: init() ^2 No QC object linked to the InFile object. Note: non-operational processing only.	Instance of c++ class "QC" does not exist (see comment of exception 1). If this is the case, exception 2 should have been thrown before.
21 VARI	from InFile.cpp: scanNcVars() Neither explicit variable name nor CMOR encoded filename. Note: search priority: clear excluded variables, time dependence, multi-dimensional variables, independent variables matched with filename.	In the setup file a fixed field variable was spcified but not found in the data.
22 VARI	from InFile.cpp: scanNcVars() Fixed field variable name from filename not found in the netCDF file.	

28 GENE	from Parse.cpp: convertEmbeddedObj() Undefined specification in the command-line arguments. Note: non-operational processing only.	Fatal, as this should not happen in the CMIP5 project. This routine is called from a script. So its command line input should be clean.
29 GENE	from Parse.cpp: convertEmbeddedObj() Syntax error in the command-line arguments. Note: non-operational processing only.	Fatal, as this should not happen in the CMIP5 project. This routine is called from a script. So its command line input should be clean.
30 OBSO		Checks from 30 to 39 refer to the consistency of parent-child relations between experiments.
30_1 ACCE	from qcExecutor_FS (script): checkParent() Invalid path to parent: <path> Note: issues a warning.	The information on the parent-child relation is requested in CMIP5 for all experiments but piControl. There the attribute should be set to n/a.
30_2 ACCE	from qcExecutor_FS (script): checkParent() No parent file found in path: <file> Note: issues a warning.	The information on the parent-child relation is requested in CMIP5 for all experiments but piControl. There the attribute should be set to n/a.
30_3 META	from QC.cpp: QC.checkCMIP5_Filename() Missing netCDF attribute: parent_experiment_id.	The information on the parent-child relation is requested in CMIP5 for all experiments but piControl. There the attribute should be set to n/a.
30_4 META	from QC.cpp: QC.checkCMIP5_Filename() Missing netCDF attribute: parent_experiment_rip.	The information on the parent-child relation is requested in CMIP5 for all experiments but piControl. There the attribute should be set to n/a. You may want to set PARENT_EXP_RIP=none in the setup file to suppress this check (e.g., for older data from when this attirbute was not yet required).
30_5 GENE	from testParentChild.cpp: testParentChild.main() Missing command-line parameter for the child experiment: -c file.	The information on the parent-child relation is requested in CMIP5 for all experiments but piControl.
30_6 GENE	from testParentChild.cpp: testParentChild.main() Missing command-line parameter for the parent experiment: -p file.	The information on the parent-child relation is requested in CMIP5 for all experiments but piControl.
31_1 ACCE	from testParentChild.cpp: testParentChild.main() Could not open the netCDF file of the child experiment	Missing authorisation.
31_2 ACCE	from testParentChild.cpp: testParentChild.main() Could not open the netCDF file of the parent experiment	Missing authorisation.
32 TIME	from testParentChild.cpp: testParentChild.getTimeProperties() Different calendar types in parent and child. Note: This could work. Thus, only a warning.	
33_1 META	from testParentChild.cpp: testParentChild.getTimeProperties() No time units attribute in the child file.	
33_2 META	from testParentChild.cpp: testParentChild.getTimeProperties() No time units attribute in the parent file.	
34_1 META	from testParentChild.cpp: testParentChild.getTimeProperties() Time units attribute: different measuring in child and parent.	
34_2 META	from testParentChild.cpp: testParentChild.getTimeProperties() Time units attribute: different reference dates in child and parent.	
35 TIME	from testParentChild.cpp: testParentChild.sync() Child begins earlier than the parent Note: potentially different time units are taken into account. The last lag of the parent is the one closest before the first one of the child.	
36 TIME	from testParentChild.cpp: testParentChild.checkTime() The lag across files differs from the last lag in the parent. Note: potentially different time units are taken into account. The last lag of the parent is the one closest before the first one of the child.	
37 TIME	from testParentChild.cpp: testParentChild.checkTime() The last lag of the parent differs from the first lag of the child. Note: potentially different time units are taken into account. The last lag of the parent is the one closest before the first one of the child.	
38_1 TIME	from testParentChild.cpp: testParentChild.checkTime() No bounds attribute of time of the parent.	
38_2 TIME	from testParentChild.cpp: testParentChild.checkTime() No bounds attribute of time of the child.	
38_3 TIME	from testParentChild.cpp: testParentChild.checkTime() Time bound lags of the last parent lag and the first lag of the child overlap.	
41 ACCE	from testParentChild.cpp: CMIP5.openTable() ^3 Could not open the CMIP5 table.	Refers to the CMIP5 standard table for comparison of variable names etc.

42 TABL	from testParentChild.cpp: CMIP5.findNextVariableHeadline() ^3 MIP table name not found in the CMIP5 table.	
43 TABL	from testParentChild.cpp: QC.checkStandardTable() ^3 Missing column(s) in the standard table.	
44 TABL	from testParentChild.cpp: QC.checkStandardTable() ^3 Variable not found in the standard table.	The variable does not need to be in the CMIP5 standard table in case option "TABLE_VARIABLE_CONSTRAINT=RELAXED" is set in the setup file. An additional variable of surface pressure (e.g., ps) to base the vertical coord system on is ok.
45_1 TIME	from testParentChild.cpp: QC.getDimMetaData() Variable time has no unit attribute.	Fatal, as the time axis cannot be checked.
45_2 TIME	from testParentChild.cpp: QC.getDimMetaData() Time attribute declares time_bounds, but there is no such variable.	
45_3 TIME	from testParentChild.cpp: QC.getVariableMetaData() Variable name in filename does not match any variable in the file.	
45_4 TIME	from testParentChild.cpp: exploitFormulaTerms() Attribute formula or formula_terms is missing for variable <var>.	
46 META	from testParentChild.cpp: CMIP5.checkGlobalAttributes() Missing required netCDF global attribute: <name>	The following exceptions (46_x) refer to the file name syntax and to NetCDF attributes. Checks are based on the conventions used in CMIP5.
46_1 META	from testParentChild.cpp: checkCMIP5_Filename() Missing netCDF attribute: project_id.	Fatal, as many parts of the CMIP5 workflow depend on the DRS syntax.
46_2 META	from testParentChild.cpp: checkCMIP5_Filename() Missing netCDF attribute: physics_version.	Fatal, as "no sub-table named in file header" should not happen when data is written by CMOR2.
46_3 META	from testParentChild.cpp: checkCMIP5_Filename() Missing netCDF attribute: initialization_method.	
46_4 META	from testParentChild.cpp: checkCMIP5_Filename() Missing netCDF attribute: realization.	The names of the CMIP5's subtable differ in file name and file header of the NetCDF file. Fatal, as this should not happen when data is written by CMOR2.
46_5 META	from testParentChild.cpp: checkCMIP5_Filename() Missing netCDF attribute: experiment_id.	
46_6 META	from testParentChild.cpp: CMIP5.checkCMIP5_Filename() Filename is inconsistent with CMOR encoding.	For the required structure of the file name see *Taylor et al, CMIP5 Data Reference Syntax (DRS) and Controlled Vocabularies*.
46_7 META	from testParentChild.cpp: CMIP5.checkCMIP5_Filename() Filename does not match CMIP5 attributes.	The file (chunk) name constructed from the header information in the file does not match the actual file name. E.g., the header's global attribute for the realisation's number (or initialisation or physics) does not match the *_r<i>i<j>p<k>_* portion of the file name.
46_8 META	from testParentChild.cpp: CMIP5.getMIP_table() Invalid MIP table name in CMIP5 attributes.	Mind: a string like "amon" is case sensitive.
47 TABL	from testParentChild.cpp: QC.checkDimTableEntry() ^5 Table error for dimension(s) of a variable.	The following exceptions (47_x) refer to the dimension information in the NetCDF file header. At first, this is compared to the CMIP5 standard table, from the second temporal subset (chunk, file) on to the internal project table.
47_1 TABL	from testParentChild.cpp: QC.checkDimTableEntry() ^5, ^7 Standard/Project table: <variable>, <dimension>: output name.	The output name is the variable acronym used in the file name.
47_2 TABL	from testParentChild.cpp: QC.checkDimTableEntry() ^5, ^7 Standard/Project table: <variable>, <dimension>: standard name.	
47_3 TABL	from testParentChild.cpp: QC.checkDimTableEntry() ^5, ^7 Standard/Project table: <variable>, <dimension>: long name.	
47_4 TABL	from testParentChild.cpp: QC.checkDimTableEntry() ^5, ^7 Standard/Project table: <variable>, <dimension>: axis.	Correlation lon/lat/alt/time vs X/Y/Z/T according standard table did not match.
47_5 TABL	from testParentChild.cpp: QC.checkDimTableEntry() ^5, ^7 Standard/Project table: <variable>, <dimension>: checksum. Note: Indicates a change in the layering or model grid.	Checksum is over the set of values of this (1D) axis, e.g., over var=altitudeValues() or var=listOfBasins:{atlantic, pacific, indic}. The axes' values of different chunks (including char dimensions) are compared by checksums to a: each other (file to file) and b: the required values of Taylor's tables (if given there). This error is "nearly fatal". However, we might accept

		obvious rounding errors.
47_6 TABL	from testParentChild.cpp: QC.checkDimTableEntry() ^5, ^7 Standard/Project table: <variable>, <dimension>: units.	
47_7 TABL	from testParentChild.cpp: QC.checkDimTableEntry() ^5, ^7 Standard/Project table: <variable>, <dimension>: bounds. a) Conflict in request for bounds. b) Bounds not available in the file.	Existence of bounds in the file header needs to be one of {yes,no}. Value found does not match.
47_8 TABL	from testParentChild.cpp: QC.checkDimTableEntry() ^5, ^7 Standard/Project table: <variable>, <dimension>: size.	Length of axis (=number of points) should be constant! Except: the number of vertical levels may vary (dim=plev).
47_9 TIME	from testParentChild.cpp: QC.checkDimTableEntry() ^5, ^7 Standard/Project table: time, units: missing key string in the units attribute 'PERIOD since'. Note: PERIOD indicates key: minutes, hours, days, etc. Note: Throws always an error flag, because the date at each time step cannot be determined later.	
47_10 TIME	from testParentChild.cpp: QC.checkDimTableEntry() ^5, ^7 Standard/Project table: time, units: different reference dates. Note: Throws always a warning flag, because the date at each time step might be found correct later.	For files where time is coded as <referenceTime>+ <timeStep>: change of reference time. This may be ok in case <timeStepUnit> is changed accordingly.
47_11 TABL	from testParentChild.cpp: QC.checkStandardTableDimBounds() ^5, ^7 Number of dim_bounds do not match those of the standard table.	
47_12 TABL	from testParentChild.cpp: QC.checkStandardTableDimBounds() ^5, ^7 Values of dim_bounds do not match those of the standard table.	
47_13 TABL	from testParentChild.cpp: QC.checkDimTableEntry() ^5, ^7 File: coordinate attribute variable: incorrect units.	
48 TABL	from testParentChild.cpp: QC.checkDimStandardTable() ^3, ^4 Dimension not found in the standard table.	
49 TABL	from testParentChild.cpp: QC.checkDimStandardTable() ^3 Table sheet for dimensions not found in the CMIP5 table.	
50 TABL	from testParentChild.cpp: QC.checkDimStandardTable() ^3 Corrupt standard sub-table for dimensions: wrong number of columns.	
51 TABL	from testParentChild.cpp: QC.checkDimStandardTable() ^3 Missing value in MIP table 'dims' in column 'bounds'?. Note: Required is 'yes' or 'no'.	
52 TABL	from testParentChild.cpp: QC.checkDimStandardTable() ^3, ^4 Dimension from the table not found in the file.	
54 ACCE	from testParentChild.cpp: QC.checkProjectTable() Could not create project table. Note: Due to a faulty configguration file entry. Note: Signal to qcManager to shutdown.	
55 TABL	from testParentChild.cpp: QC.checkProjectTable() ^2 Corrupt project table: variable | dimension | auxiliary.	
56 TABL	from testParentChild.cpp: QC.checkProjectTableAuxiliary() Project-table errors for auxiliaries.	The following exceptions (56_x) refer to checks of auxiliary variables in the file against the internal project table. Here "auxiliary variables" refers to all time independent variables (except a fixed variable as standard name), e.g. dimensions etc.
56_1 TABL	from testParentChild.cpp: QC.checkProjectTableAuxiliary() Project-table: time independence check failed.	A variable which changed from being independent on time to being time dependent.
56_2 TABL	from testParentChild.cpp: QC.checkProjectTableAuxiliary() Project-table: number of dimensions of auxiliary changed.	E.g., a change in the number of pressure levels.
56_3 TABL	from testParentChild.cpp: QC.checkProjectTableAuxiliary() Project-table: checksum of levels or grid values changed.	Checksum of the set of values of a space dimension has changed between files.
56_4 TABL	from testParentChild.cpp: QC.checkProjectTableAuxiliary() Project-table: missing auxiliary in the file.	An auxiliary variable "disappeared" between two temporal subsets (chunks, files).
56_5 TABL	from testParentChild.cpp: QC.checkProjectTableAuxiliary() Project-table: missing auxiliary in the table.	An auxiliary variable "appeared" between two temporal subsets (chunks).
56_6 TABL	from testParentChild.cpp: QC.getVariableMetaData() Option AUXILIARIES is not syntax compliant.	
57 ACCE	from testParentChild.cpp: QC No path to tables; faulty config-file entry. Note: Signal to qcManager to shutdown.	No path to CMIP5 standard table OR to project table ==> no checks possible ===> end of work.

58 TABL	from testParentChild.cpp: QC.checkVarTableEntry() ^5 Table error for a variable.	The following exceptions (58_x) refer to checks of the main variable in the file against the internal project table (or the CMIP5 standard table). You may use a flag in the setup file to switch between the detailed code (58_x) and the lumped coarse form (just "58").
58_1 TABL	from testParentChild.cpp: QC.checkVarTableEntry() ^5 Standard/Project table: variable: standard name conflict.	Different standard names in NetCDF header and project/standard table.
58_2 TABL	from testParentChild.cpp: QC.checkVarTableEntry() ^5 Standard/Project table: variable: long name conflict.	Different long names in NetCDF header and project/standard table.
58_3 TABL	from testParentChild.cpp: QC.checkVarTableEntry() ^5 Standard/Project table: variable: units conflict. a) variable has no units attribute, table requires units= or units=1. b) variable has no units attribute. c) variable has <v_units>, table requires <t_units>.	Different units in NetCDF header and project/standard table.
58_4 TABL	from testParentChild.cpp: QC.checkVarTableEntry() ^5 Standard/Project table: variable: cell-method conflict.	Different cell methods in NetCDF header and project/standard table. Cell method may be missing in the file header (or the table) or is wrong. In the latter case, this will affect the time step width (provided it refers to dimension time) and will be detected as time axis inconsistency.
58_5 TABL	from testParentChild.cpp: QC.checkVarTableEntry() ^5 Standard/Project table: variable: type conflict.	Different variable type (float expected) in NetCDF header and project/standard table. Fatal, as this variable will not be checked.
58_6 TABL	from testParentChild.cpp: QC.checkSigmaPressureCoordinates() No variables a(k) or b(k) found for the hybrid sigma pressure coordinates.	
58_7 TABL	from testParentChild.cpp: QC.checkVarTableEntry() ^5 Standard/Project table: variable: cell-measures conflict.	Dimensions in CF carry the "cell-measure" attribute (mean,..). This information is compared to the entries in the Taylor tables.
58_8 TABL	from testParentChild.cpp: QC.checkVarTableEntry() ^5 Standard/Project table: variable: incorrect coordinate attribute.	
58_9 TABL	from testParentChild.cpp: QC.checkVarTableEntry() ^5 Standard/Project table: variable: type check discarded, not specified in the MIP Table.	The type of the variable could not be checked as it was not found in the MIP table.
59 VARI	+	The following exceptions (59_x) refer to checks of variables.
59_1 VARI	from testParentChild.cpp: QC.finally() Data set entirely of const value.	
59_2 VARI	from testParentChild.cpp: QC.finally() Data set entirely of _FillValue.	
59_3 VARI	from testParentChild.cpp: QC.finally() All records (data at given time steps) are identical.	
60 META	from testParentChild.cpp: QC.initResumeSession() Name of a variable has changed Note: Compared to previous chunks.	
63 OBSO	from testParentChild.cpp: QC.sync() All records have previously been processed. Note: This is not an error: idle	All existing records of the file have been processed. However, expected file end (according to file name) and time stamp of last record do not match. This should not happen, if you run these checks on a non-increasing (=not supplemented) fixed set of data.
64 GENE	from testParentChild.cpp: QC.syncRedo() Indication of a renewal of the input file.	This should not happen unless you run QC during the model run. In this case it means that the model perhaps has newly started to write this file (i.e., less time steps as during last visit of this file were found by the checking routine).
65 GENE	from testParentChild.cpp: QC.syncRedo() Faulty argument in config-file for redo-option.	
66 GENE	from testParentChild.cpp: QC.syncRedo() REDO is selected, but the qc-file is empty.	
67 GENE	from testParentChild.cpp: QC.syncRedo() REDO requires at least one record in qc_<filename>.nc	
68	from testParentChild.cpp: QC.testTimeStep() Waiting for a closing temporal gap. Note: This is not an error.	tbd...

69 TIME	from testParentChild.cpp: QC.initResumeSession() ^2 Sub-cycle of time step has changed. Note: For any kind of cycles within a time step (e.g. diurnal cycle). Note: Not for CMIP5	Should not happen in CMIP5 as values of, e.g., subcycle UTC 0/6/12/18 never are aggregated to four distinct monthly means.
70 GENE	from testParentChild.cpp: QC.(different methods) Any error in any record.	In this case, F<value> is given for specification according to the table at the top of this page (see file qc_<fileName>.nc, too). The severity level of this exception depends on the level of exception F<value>.
71 GENE	from testParentChild.cpp: QC.testTimeStep() Error in a time record (Error Flags: 1, 2, 4) stops and blocks further attempts, if option STOP_AT_TIME_ERROR is enabled.	In this case, F1,F2 or F4 is given for specification according to the table at the top of this page. No further attempts to check these Atomic Dataset are made.
71_1	from testParentChild.cpp: qcManager Ambiguity check failed for sorted dates of sub-temporal file set.	This check refers to the modification time stamp which is stored by the operating system of the computer. The modification times of the files (chunks) should be ascending for files with ascending values on their time axes. In this case the check failed.
71_2	from testParentChild.cpp: qcManager Modification time check failed for ascending dates of sub-temporal files.	This check refers to the modification time stamp which is stored by the operating system of the computer. The modification times of the files (chunks) should be ascending for files with ascending values on their time axes. In case of this warning it is not. The check may be suppressed by setting "SYNC_FILE_AMBIGUITY_CHECK = NO_MOD".
72 TABL	from testParentChild.cpp: QC.parseTimeTable() Violation against time information of the standard table.	The following exceptions (72_x) refer to checks of the file time axis against the CMIP5 standard table's time information.
72_1 TABL	from testParentChild.cpp: QC.parseTimeTable() Time record before the first time-table range.	
72_2 TABL	from testParentChild.cpp: QC.parseTimeTable() Time record after the last time-table range	
72_3 TABL	from testParentChild.cpp: QC.parseTimeTable() Too many time records compared to the time-table.	
95 TIME	from testParentChild.cpp: QC.testTimeStamp() Ambiguous or faulty first time-stamp in the filename. Note: Faulty, if the stamp is younger than 1st time-record. Ambig., if the stamp is too coarse to resolve time-records.	The time of the file's start record is in conflict with the starttime of the file name. The max allowed difference depends on the step width.
96 TIME	from testParentChild.cpp: QC.testTimeStamp() Filename time-stamp error (2nd date). Note: Time record exceeds time-stamp of the filename.	The time of a file's record is not in the time span of the file name.
97 TIME	from testParentChild.cpp: QC.testTimeStamp() Invalid time-range in filename.	The file name does not provide a legal (:= CMOR2 conform) start-stop time span (start>stop).
99 GENE	from testParentChild.cpp: QC.finally() Status: apparently in progress. Note: This indicates that no error occurred. But, the end time of the input filename and the last time value in the data did not match (with the uncertainty of 0.75 of a time step). So, it is likely that the file is still in a stage of processing.	As these procedures are meant to run on continuously supplemented files, <stop time of file name> != <time of last record> might indicate that this file has not yet been readily filled.
111 GENE	from testParentChild.cpp: qc-main The program exited without any exit code above and did not produce a single file. This is an error.	A run that has not found any exception should have left a NetCDF file.

Footnotes:

1. also across from previous file
2. Non-operative mode.
3. for option TABLE_STANDARD=... set in the configuration
4. for option TABLE_RELAXED_DIM_CONSTRAINT disabled
5. a series of tests where a failure of one of a few of them aborts the program and leads to a blocking of further attempts.
6. writes messages to a file qc_error_<variable...>.nc and blocks further attempts
7. conflict is reported, if the attribute of the dimension is available in both table and file meta data. If one is missing, this is reported.