Gaia Parameter Database: contents maintenance

Jos de Bruijne
Astrophysics & Fundamental Physics Missions Division (SCI–SA)
SCI–SA / RSSD / ESTEC / ESA
Postbus 299
NL–2200 AG Noordwijk (ZH)
The Netherlands
Jos.de.Bruijne@rssd.esa.int

GAIA-CA-SP-ESA-JDB-027-1
July 13, 2006

Abstract This document describes how to manage the scientific contents of the Gaia Parameter Database. Together with the user manual (GAIA-UL-001) and the document on the technical implementation of the database (in preparation by U. Lammers), these three documents provide a full overview of all aspects of the database. Any questions on the database may be sent to `gaialib@rssd.esa.int`.

1 Introduction

Some background on and features of the Gaia Parameter Database are described in Gaia-JdB-007 (revision 1, 6 June 2003) and in pages 67–70 of ESA SP–576 (proceedings of the symposium 'The three-dimensional universe with Gaia' held in 2004 in Paris–Meudon). A user manual of the database is available in GAIA-UL-001 (revision 1.3, 27 April 2004). A document describing implementation details behind the database is in preparation. This document covers a more administrative aspect of the Gaia Parameter Database, namely how to manage its scientific contents. This type of maintenance covers several aspects:

- Updating parameters;
- Adding new parameters;
- Deleting existing parameters;
- Renaming parameters;
- Handling multi-dimensional parameters;
- Testing database updates;
- Updating the on-line database;
- New database releases.

These aspects will be described below, after first providing a more general introduction.

2 General set-up

The database, from the contents point of view, is represented by:

- A single, user-modifiable database contents file, here refered to as `gaiaparam_db_tables.txt`. For the Gaia set-up, this file is located in <`HOME`>`/paramdb/contents/` of the gaialib account. For the Gaia set-up, the home directory is <`HOME`> = `/usr4/users/gaialib/` on the PHP-enabled web server `rssd-svr.estec.esa.int`;
- A ChangeLog file in which parameter and database-version changes can be accumulated/summarised. This file, called `ChangeLog.xml` and located in <`HOME`>`/paramdb/contents/` for the Gaia set-up, is visible from the database website;
- An FTP directory structure — with root <`FTP-ROOT`> — containing FITS files which represent the multi-dimensional parameters in the database. For the Gaia set-up, the FTP root directory is <`FTP-ROOT`> = `/services/ftp/pub/Gaia/data/CalParam/` = `/usr4/users/gaialib/pubftp/-data/CalParam/`.

The Gaia Parameter Database, in essence, consists of a series of `mysql` database tables which can be queried using a web interface. The contents administrator never touches these database tables directly in the `mysql` database but always uses a single database upload file, here refered to as <`HOME`>`/paramdb/contents/gaia param_db_tables.dat`. This upload file, in turn, is never touched (edited) directly by the contents administrator but is always obtained from the main contents file (<`HOME`>`/paramdb/contents/gaiaparam_db_ tables.txt`; Section 3) by execution of a stand-alone Fortran-77 executable called <`HOME`>`/paramdb/-contents/gaiaparam_db_tables.e`. Details of this conversion are given in Section 9. The contents file, the upload file, and the Fortran code (source and executable) are all located in <`HOME`>`/paramdb/contents/`.

3 Contents file

The database contents file (<`HOME`>`/paramdb/contents/gaiaparam_db_tables.txt`) is the only entry point for the database administrator for making contents changes to the database. The contents file contains, in broad terms:

- A header, followed by;
- One dedicated line for each parameter in the database.

The header contains some comment lines, identified with '#' signs, and one initialisation line. This line looks like:

NULL| NULL| NULL| NULL| NULL| NULL| NULL| NULL| NULL | | ' '| ' '| ' '| ' '| ' '| ' '| ' '| ' '| ' '| ' '| '2006-03-01T09:00:00'.

The initialisation line should not be modified, unless additional fields/columns are introduced in the database.

The parameter lines contain a fixed number of fields/columns, using a '|' sign as field separator. Strings are enclosed within '...' symbols. An example of a parameter line is:

Nature| | | | Pi| Constant| | | CONF| 0| '3.141592653589793238462643383 28'| '%s'| 'F'| 'basic'| 'scalar'| ' '| 'The constant Pi (also known as Archimedes\' constant)'| 'Well-known mathematical constant; numerical value can be extracted, e.g., from Mathematica 4.0 for Solaris (Wolfram Research, Inc.) using \'N[Pi,30]\' '| ' '| '\pi'| '2006-03-01T09:00:00'.

The 21 following fields/columns shall be present for each parameter in its associated parameter line (the example above is expanded below, for clarification):

1. Level-1 identifier = Nature. This is `mysql` table `SCOPES0`. This level may *not* be VOID.
2. Level-2 identifier = VOID. This is `mysql` table `SCOPES1`. This level may be VOID.
3. Level-3 identifier = VOID. This is `mysql` table `SCOPES2`. This level may be VOID but if it is non-VOID, the level-2 identifier should also be set/defined.
4. Level-4 identifier = VOID. This is `mysql` table `SCOPES3`. This level may be VOID but if it is non-VOID, the level-2 and -3 identifiers should also be set/defined.
5. Class identifier = Pi. This is `mysql` table `CLASSES`. This entry may *not* be VOID.
6. Kind identifier = Constant. This is `mysql` table `KINDS`. This entry may be VOID.
7. Direction identifier = VOID. This is `mysql` table `DIRECTIONS`. This entry may be VOID. Only allowed other entries are AL and AC (without quotes). These acronyms/abbreviations refer to ALong-scan and ACross-scan, which are Gaia-specific denominations. However, one may imagine replacing these terms by X and Y (and adding Z, if needed).
8. Case identifier = VOID. This is `mysql` table `CASES`. This entry may be VOID.
9. Status identifier = CONF. This is `mysql` table `STATUS`. This field is generally refered to as 'Status'. This entry may *not* be VOID. Proposed entries are CONF, TBC, etc., but any other value is in principle allowed, provided the string length is limited to 12 characters.
10. Value identifier part 1 = 0. Entry must be a number, either integer or float.
11. Value identifier part 2 = '3.14159265358979323846264338328'. Entry must be a string (empty string ' ' allowed).
12. Format identifier = '%s'. C-type format identifier (e.g., '%s', '%.1e', '%.8f', '%d', etc.; see `http://en.wikipedia.org/wiki/C_programming_language` and `http://en.wikipedia.org/wiki/Printf` for an overview). The format string shall be limited to 64 characters.
13. Parameter type = 'F'. Allowed entries are: 'B', 'D', 'E', 'F', 'I', 'J', 'S', 'T'. These parameter types are defined below, under Comment 9, items (a)– (l).
14. Basic or derived = 'basic'. This (boolean) field is generally refered to as 'Basic'. Allowed entries are 'basic' and 'derived'.
15. Scalar or multi-dimensional = 'scalar'. This (boolean) field is generally refered to as 'Scalar'. Allowed entries are 'scalar' and 'multi'.
16. Unit = ' '. This field is generally refered to as 'Unit'. Entry must be a string (empty string ' ' allowed). Examples are: 'mm s ˆ -1', 'mag at 1 e ˆ - s ˆ -1', 'mas', etc. The unit string shall be limited to 64 characters.
17. Description = 'The constant Pi (also known as Archimedes\' constant)'. Entry must be a string (empty string ' ' allowed). This field is generally refered to as 'Description'.

18. Source = 'Well-known mathematical constant; numerical value can be extracted, e.g., from Mathematica 4.0 for Solaris (Wolfram Research, Inc.) using \'N[Pi,30]\' '. Entry must be a string (empty string ' ' allowed). This field is generally refered to as 'Source'.

19. Expression = ' '. Entry must be a string. This field is generally refered to as 'Expression'. If a parameter is 'basic', the string must be empty (' '). For 'derived' parameters, an expression must be provided. Examples are: '%:Satellite:AF:Sky_NumberOfPhotoElectrons% * %:Satellite:CCD_ExposureTime_Effective[11]% / %:Satellite:CCD_ExposureTime%', 'pow(%Optics_DioptricElementTransmissivity_860nm%, 2 * %Optics_NumberOfDioptricElements%) * %GratingPlate_Transmissivity_860nm% * %FilterTransmissionCurve_Transmissivity_860nm%', etc.

20. LaTeX label = '\pi'. Entry must be a string (empty string ' ' allowed). This field is generally refered to as 'LaTeX label'. The LaTeX-label string shall be limited to 255 characters.

21. Last modified = '2006-03-01T09:00:00'. Entry must be a string. This field is generally refered to as 'Last modified'. It is used to allow selection of recently modified parameters. The date/time format as shown in the example must be adhered to ('YYYY-MM-DD T HH:MM:SS', but without internal blanks around the middle T). This field must be the final field.

Some comments on the above 21 fields:

1. The construct ':Level-1:Level-2:Level-3:Level-4:CLASS_KIND_DIRECTION_CASE' is generally refered to as 'Name'. Names must be unique among all parameters. It is the administrator's responsibility to ensure that names *are* unique; no intelligent software checking on name-uniqueness has been implemented.

2. Full parameter names, as defined above, can be decomposed into two parts: (a) the hierarchical level, followed by (b) the basename of the parameter name:

   (a) Hierarchical levels are separated by ':' signs. The full hierarchical level always starts with a ':' sign and is also always separated from the basename of the parameter name by a ':' sign.

   (b) In case one or more of the optional fields KIND, DIRECTION, and/or CASE is set, an underscore ('_' sign) shall be used as (internal) separator[1].

3. Spaces (blanks) in the contents file are transparent (except within strings) and may be used to outline the various fields in the contents file visually.

4. The use of a quote within a string is possible by escaping it using a backslash ('\' sign).

[1]Note the following detailed caveat/requirement resulting from the freedom to use KIND and CASE as independent optional extensions to CLASS: it shall be ensured that the ensemble collection — taking all parameters in the database into account — of all entries used for KIND and the ensemble collection — again taking all parameters in the database into account — of all entries used for CASE shall not have any common entries. This requirement is needed in order to avoid ambiguous parameter names arising internally in the database software environment. Although these ambiguities are not necessarily directly visible to the user (they depend on the precise interactions between all parameters in the database), they are potentially dangerous and shall be avoided. For example, the simultaneous presence in the database of two parameters with basenames CLASS_KIND_CASE = Performance_Astrometry_Requirement and CLASS_KIND_CASE = DataCompression_Ratio_Astrometry is not allowed, since KIND = Astrometry for the first parameter name and CASE = Astrometry for the second parameter name. It is the administrator's responsibility to ensure that KIND and CASE have no common entries in the database. No intelligent software checking on this has been implemented.

5. Empty lines shall not be present in the contents file (have them start with '#' signs instead). The presence of one or more empty lines causes the Fortran executable to terminate abnormally with an error message (or simply crash, depending on the compiler).

6. Within expressions, parameter names shall be enclosed within '%' signs. The mathematical-operator set from PHP (identical in practice to the one from ANSI-C) shall be used within expressions (for example, using 'pow(x,y)' for 'x^y'; see `http://www.wiki.cc/php/Main_Page` for an overview).

7. Vector elements can be addressed in expressions using '[i]' constructs, where $i = 0, \ldots, n-1$ denotes the vector element for a vector of length n.

8. If level identifiers are (or if the full hierarchical level is) omitted within an expression which is used to define a derived parameter, the input parameter in the expression is automatically searched for (and assumed to be available at) at the *same* hierarchical level as which the derived parameter is being defined at. Consider, for instance, the case where a derived parameter :A:B:C_D is defined in the database as the product of two other parameters, namely :E:F_G and :A:B:H_I. The expression for :A:B:C_D could then either read :A:B:C_D = '%:E:F_G% * %:A:B:H_I%' or, equivalently and shorter, :A:B:C_D = '%:E:F_G% * %H_I%'. The latter option, without specification of a hierarchical level of the input parameter H_I, is equivalent to the first option, where the input parameter is addressed using the full level as :A:B:H_I, because the database software automatically assumes H_I is a parameter which is defined at the *same* hierarchical level as the parameter that is being defined in the expression, :A:B: from :A:B:C_D in this example.

9. Eight parameter types are available, namely 'B', 'D', 'E', 'F', 'I', 'J', 'S', and 'T'. These letters provide 11 distinct possibilities (words typeset in CAPITAL font in the list below refer to user-defined entries, for example VALUE can be 3.5, EXPRESSION can be %:Satellite:Telescope_Transmissivity_860nm% * %Optics_Transmissivity_860nm% * %CCD_QE_160K860nm%, etc.):

   (a) Basic (non-derived) integer-type scalar parameters.
      - Parameter type = 'I'.
      - Format identifier = '%d'.
      - Value identifier part 1 = VALUE.
      - Value identifier part 2 = ' '.
      - Basic or derived = 'basic'.
      - Scalar or multi-dimensional = 'scalar'.
      - Expression = ' '.

   (b) Derived integer-type scalar parameters.
      - Parameter type = 'I'.
      - Format identifier = '%d'.
      - Value identifier part 1 = 0.
      - Value identifier part 2 = ' '.
      - Basic or derived = 'derived'.
      - Scalar or multi-dimensional = 'scalar'.
      - Expression = 'EXPRESSION'.

   (c) Basic (non-derived) float-type scalar parameters.
      - Parameter type = 'D'.
      - Format identifier = '%f' or '%e'.

- Value identifier part 1 = VALUE.
- Value identifier part 2 = ' '.
- Basic or derived = 'basic'.
- Scalar or multi-dimensional = 'scalar'.
- Expression = ' '.

(d) Derived float-type scalar parameters.

- Parameter type = 'D'.
- Format identifier = '%f' or '%e'.
- Value identifier part 1 = 0.
- Value identifier part 2 = ' '.
- Basic or derived = 'derived'.
- Scalar or multi-dimensional = 'scalar'.
- Expression = 'EXPRESSION'.

(e) Basic (non-derived) integer-type vector parameters.

- Parameter type = 'J'.
- Format identifier = '%s'.
- Value identifier part 1 = 0.
- Value identifier part 2 = 'VECTOR ELEMENT 0, VECTOR ELEMENT 1, ..., VECTOR ELEMENT $n-1$'.
- Basic or derived = 'basic'.
- Scalar or multi-dimensional = 'scalar'.
- Expression = ' '.

(f) Derived integer-type vector parameters.

- Parameter type = 'J'.
- Format identifier = '%d'.
- Value identifier part 1 = 0.
- Value identifier part 2 = ' '.
- Basic or derived = 'derived'.
- Scalar or multi-dimensional = 'scalar'.
- Expression = 'EXPRESSION FOR VECTOR ELEMENT 0, EXPRESSION FOR VECTOR ELEMENT 1, ..., EXPRESSION FOR VECTOR ELEMENT $n-1$'.

(g) Basic (non-derived) float-type vector parameters.

- Parameter type = 'E'.
- Format identifier = '%s'.
- Value identifier part 1 = 0.
- Value identifier part 2 = 'VECTOR ELEMENT 0, VECTOR ELEMENT 1, ..., VECTOR ELEMENT $n-1$'.
- Basic or derived = 'basic'.
- Scalar or multi-dimensional = 'scalar'.
- Expression = ' '.

(h) Derived float-type vector parameters.

- Parameter type = ’E’.
- Format identifier = ’%f’ or ’%e’.
- Value identifier part 1 = 0.
- Value identifier part 2 = ’ ’.
- Basic or derived = ’derived’.
- Scalar or multi-dimensional = ’scalar’.
- Expression = ’EXPRESSION FOR VECTOR ELEMENT 0, EXPRESSION FOR VECTOR ELEMENT 1, ..., EXPRESSION FOR VECTOR ELEMENT $n - 1$’.

(i) Basic (non-derived) float-type parameters for which the representation and definition requires special care regarding floating-point rounding etc. Currently used for two parameters, namely $\pi = 3.14...$ and e = 2.71....

- Parameter type = ’F’.
- Format identifier = ’%s’.
- Value identifier part 1 = 0.
- Value identifier part 2 = ’VALUE’.
- Basic or derived = ’basic’.
- Scalar or multi-dimensional = ’scalar’.

(j) Basic (non-derived) string-type parameters. Derived strings do not exist.

- Parameter type = ’S’.
- Format identifier = ’%s’.
- Value identifier part 1 = 0.
- Value identifier part 2 = ’VALUE’.
- Basic or derived = ’basic’.
- Scalar or multi-dimensional = ’scalar’.

(k) Basic (non-derived) boolean parameters. Derived booleans do not exist.

- Parameter type = ’B’.
- Format identifier = ’%s’.
- Value identifier part 1 = 0 or 1 (0 represents false; 1 represents true).
- Value identifier part 2 = ’ ’.
- Basic or derived = ’basic’.
- Scalar or multi-dimensional = ’scalar’.

(l) Basic (non-derived) multi-dimensional parameters. Derived multi-dimensional parameters do not exist.

- Parameter type = ’T’.
- Format identifier = ’%s’.
- Value identifier part 1 = 0.
- Value identifier part 2 = ’%SELF%_NUMBER.fits’.
    i. The %SELF% is a mandatory literal entry, not to be changed by the administrator. The %SELF% is expanded automatically to the appropriate FITS filename in the FTP directory structure. For example, with a parameter name :Satellite:SM:CCD_QE_Requirement160K, the associated FITS file link is automatically generated to be `<FTP-ROOT>/Satellite/SM/CCD_QE_Requirement160K_001.fits`. See Section 8 for details.

ii. The NUMBER is a three-digit entry, including leading zeroes, which refers to the version of the FITS file (e.g., 001 for the first FITS-file version). New versions of FITS files can be addressed / made available to on-line users by manually updating the NUMBER counter by 1 (from 001 to 002 in the example here). Again, Section 8 contains further details.

- Basic or derived = 'basic'.
- Scalar or multi-dimensional = 'multi'.

4 Updating parameters

In order to update an existing parameter, load the contents file (`<HOME>/paramdb/contents/gaiaparam_db_tables.txt`) into your editor. Then make the change, for example update the value-identifier field from 5.46 to 5.78, or update the description, or add a LaTeX label, etc. Then modify the last-modified field to today's date and time. For actually updating the on-line database (contents) and making the change effective, refer to Section 11.

5 Adding new parameters

In order to add a new parameter to the database, load the contents file (`<HOME>/paramdb/contents/gaiaparam_db_tables.txt`) into your editor. Add, at an arbitrary location (but after the header), a new line and fill/define all 21 fields. Verify that name uniqueness for ':Level-1:Level-2:Level-3:Level-4:CLASS_KIND_DIRECTION_CASE' is ensured since no software check is made on this. For actually updating the on-line database (contents) and making the change effective, refer to Section 11.

6 Deleting parameters

In order to delete an existing parameter from the database, load the contents file (`<HOME>/paramdb/contents/gaiaparam_db_tables.txt`) into your editor. Remove the entire line corresponding to the parameter from the file (do not leave an empty line). For actually updating the on-line database (contents) and making the change effective, refer to Section 11. Note: it is mandatory to document parameter deletions in the ChangeLog file (this is not automated).

7 Renaming parameters

In order to rename an existing parameter in the database, load the contents file (`<HOME>/paramdb/-contents/gaiaparam_db_tables.txt`) into your editor. Locate the parameter of interest and change the name using the ':Level-1:Level-2:Level-3:Level-4:CLASS_KIND_DIRECTION_CASE' fields. Verify that name uniqueness remains ensured since no software check is made on this. Change the last-modified field to today's date and time. For actually updating the on-line database (contents) and making the change effective, refer to Section 11. Note: it is mandatory to document name changes in the ChangeLog file (this is not automated).

8 Multi-dimensional parameters

Multi-dimensional parameters have a special status, also because a strong manual component unavoidably goes with the maintenance of these parameters. Effectively, each multi-dimensional parameter requires the availability of several items/files[2]:

- A regular, one-line entry in the contents file, but with the scalar/multi-dimensional field 'Scalar' set to 'multi' and other fields set according to Section 3, Comment 9, item (l). From the parameter name (e.g., :Satellite:SM:CCD_QE_Requirement160K), an auxiliary 'filename' is automatically derived in the internal database environment which we will refer to here as `<MULTIFILENAME>` (Satellite_SM_CCD_QE_Requirement160K in this example). This filename governs FITS-file creation for multi-dimensional parameters. Note that this internal 'filename' has any hierarchical level separator — meaning any ':' signs — replaced by underscores while the leading ':' sign is removed at the same time;
- A FITS-column description file `<HOME>/paramdb/contents/fits/<MULTIFILENAME>_coldes.txt` which defines the FITS-column names and FITS-data properties;
- A FITS-header file `<HOME>/paramdb/contents/fits/<MULTIFILENAME>_head1.txt` which defines the FITS-data-display and -format properties;
- An ASCII file `<HOME>/paramdb/contents/fits/<MULTIFILENAME>.txt` which defines the multi-dimensional data to be included in the FITS file (in this example, an ASCII file with two columns, one for wavelength and one for QE).

It is the administrator's responsibility to ensure that these files exist, with appropriate contents. Further details on these files are not given in this document (but see Appendix A, where some examples are provided). It is advisable to study/consult existing files (examples), the educational value of which is far larger than any written instructions that can be provided here.

With execution of the Fortran executable to process the contents file (Section 9), multi-dimensional parameters are treated distinctly from all other parameters. For each multi-dimensional parameter, a series of commands is automatically generated and stored in a script-like source file, `<HOME>/paramdb/contents/fits/gaiaparam_db_tables_multi.src` in the Gaia set-up. This source file must subsequently be executed by the administrator. Only this latter execution creates[3] the FITS files, including appropriate header information, and copies them to the FTP directory structure such that they are available for on-line users of the database.

Effectively, the following steps are taken for each multi-dimensional parameter with execution of the Fortran executable and subsequent execution of the script-like source file:

- Those contents-file fields that need to be included in the FITS-file header (currently: 'Description', 'Source', 'Expression', 'LaTeX label', and 'Last modified') are taken from the contents file and automatically transformed into a FITS-header file `<HOME>/paramdb/contents/fits/<MULTIFILENAME>_head2.txt`;
- The existing `_head1.txt` and newly-created `_head2.txt` header files are merged to a single FITS-header file `<HOME>/paramdb/contents/fits/<MULTIFILENAME>_head.txt`;

[2] In addition, an auxiliary FITS-file header file named `<HOME>/paramdb/contents/fits/gaiaparam_db_tables_comment.txt` should be available (see the Appendix for details).

[3] A mechanism has been implemented to avoid — for an administrator-defined subset of all multi-dimensional parameters — the actual creation of FITS files from ASCII files. For these files, the FITS creation is disabled, but the headers are updated. This mechanism is in place to allow external parties to deliver FITS files for inclusion in the database, but to have these files adhere to the same header-format standards as the other FITS files. The Fortran executable currently accepts up to 100 externally-provided FITS files (set by the declaration 'PARAMETER(IMULTIEXCEPTIONMAX = 100)'). The mechanism acts only on those multi-dimensional parameters which have a matching `<MULTIFILENAME>_<NUMBER>` entry in the file `<HOME>/paramdb/contents/fits/gaiaparam_db_tables_exceptions.txt`.

- The existing ASCII data `.txt` file, the existing FITS-column description `_coldes.txt` file, and the just-created FITS-header `_head.txt` file are 'combined', using the `FTOOLS` command `fcreate`, to create the FITS file <`HOME`>`/paramdb/contents/fits/`<`MULTIFILENAME`>`_`<`NUMBER`>`.fits`. As explained in footnote 3, this step can partly be by-passed for an administrator-defined subset of multi-dimensional parameters in order to cater for externally-provided FITS files;
- Three FITS-header keywords are inserted/updated (ORIGIN, TELESCOP, and INSTRUME) using the `FTOOLS` command `fparkey`;
- In order to make the FITS file publicly available, it is (secure-)copied to the FTP directory structure with root <`FTP-ROOT`>. (Hierarchical-)level identifiers in the internal 'filename' <`MULTIFILENAME`> are automatically expanded to sub-directories under <`FTP-ROOT`> (e.g.,<`HOME`>`/paramdb/contents/-fits/Satellite_SM_CCD_QE_Requirement160K_001.fits` is copied to <`FTP-ROOT`>`/Satellite/SM/CCD_QE_Requirement160K_001.fits`). In order to ensure that the appropriate sub-directories under <`FTP-ROOT`> exist, a second source file, <`HOME`>`/paramdb/contents/fits/gaiaparam_db_tables_directories.src` for the Gaia set-up, is created by the Fortran executable. This source file also needs to be executed by the administrator, prior to invoking the primary script-like source file <`HOME`>`/paramdb/contents/fits/gaiaparam_db_tables_multi.src`.

In summary, the proper procedure for maintenance of multi-dimensional parameters is as follows:

1. Make sure that the `.txt`, `_coldes.txt`, and `_head1.txt` files exist in <`HOME`>`/paramdb/contents/fits/`;
2. Make sure that the associated line in the contents file is included and up-to-date;
3. Run the Fortran executable to convert the contents file to the upload file (Section 9);
4. Execute <`HOME`>`/paramdb/contents/fits/gaiaparam_db_tables_directories.src` to ensure that all FTP sub-directories under <`FTP-ROOT`> exist;
5. Execute <`HOME`>`/paramdb/contents/fits/gaiaparam_db_tables_multi.src` to create all FITS files and copy them to the FTP area;
6. For actually updating the on-line database (contents) and making the change effective, refer to Section 11.

9 Upload file

The database upload file <`HOME`>`/paramdb/contents/gaiaparam_db_tables.dat` is always obtained from the contents file <`HOME`>`/paramdb/contents/gaiaparam_db_tables.txt` by execution of the stand-alone Fortran executable <`HOME`>`/paramdb/contents/gaiaparam_db_tables.e`. The contents administrator should never edit the upload file directly. If the upload file is incorrect, either the underlying contents file or the Fortran translator itself shall be modified.

Effectively, the Fortran executable translates the contents file from a human-readable to a database-type format. The precise translation details are irrelevant in this context, but effectively consist of mapping lists of names into unique lists, making integer-indices, cross-correlation tables, etc.

The Fortran executable takes one command-line argument, namely the filename of the upload file. As an example, the executable may be run using:

```
gaiaparam_db_tables.e gaiaparam_db_tables.dat
```

Normally, this execution will progress with messages as follows:

```
 1449 lines read from gaiaparam_db_tables.txt
 1353 parameters processed to gaiaparam_db_tables.dat
Found existing FITS file: <FITS filename 1>
Found existing FITS file: <FITS filename 2>
<etc.>
Do not forget to update the ChangeLog.xml file
```

The first message ('1449 lines read ...' in this example) displays the number of lines that was read from the contents file; the counter includes comment lines in the contents file (i.e., those lines starting with #-symbols). The second message ('1353 parameters processed to gaiaparam_db_tables.dat') informs the user of the number of parameters that was found in the contents file and that has been processed for upload in the user-defined upload file. After these two messages, a series of messages regarding FITS files is given, listing all FITS files which have been provided externally (Section 8 and footnote 3). Finally, a warning message is printed reminding the administrator that changes may need to be manually registered in the ChangeLog file (Section 11).

If the Fortran executable exits with an error (or if it crashes during execution), the input contents file contains an error. This error shall always be fixed *before* updating the database (and/or testing an upload). From experience, an easily-made error is the presence of an empty line in the contents file, which is not allowed (have such lines start with '#' signs instead).

Default settings in the Fortran translator, which may in principle be modified by the administrator if needed, are:

- Only the first 'IMAX = 5000' lines in the contents file are processed/translated. If the contents file contains more than 5000 lines (including comment lines), the value of the parameter IMAX shall be increased;
- Lines in the contents file do not have more than 4000 characters. If (some) lines in the contents file contain more than 4000 characters, the declarations 'PARAMETER (LINELENGTHMAX = 4000)', 'CHARACTER LINE*4000', 'CHARACTER CONTENTS*4000(1:IMAX)', and 'CHARACTER STRING*4000' shall be changed;
- Precisely 'KKMAX = 4' hierarchical levels are present, namely Level-1 through Level-4. These levels refer to the first 4 columns/fields in the contents file;
- The first 'KMAX = 9' parameter fields, corresponding to the first 9 columns/fields in the contents file, require indexing since (only) these fields have associated, dedicated `mysql` database tables. Currently, these are the following 9 fields: 'Level-1', 'Level-2', 'Level-3', 'Level-4', 'Class', 'Kind', 'Direction', 'Case', and 'Status' (see Section 3 for details);
- The number of fields in the contents file, separated by |-symbols, equals 'NPIPEMAX + 1 = 20 + 1 = 21'. Recall that the last field shall always be 'Last modified';

- Only the last 'NPIPEFITSLENGTH = 5' fields do appear as COMMENT lines in FITS-file headers. Currently, these are the following 5 fields: 'Description', 'Source', 'Expression', 'LaTeX label', and 'Last modified';
- The following 7 database columns/fields have maximum lengths of 64 characters (as set by the declarations 'PARAMETER (UNIQUENAMELENGTHMAX = 64)', 'CHARACTER UNIQUENAME*64-(1:KMAX,1:IMAX)', and 'PARAMETER (UNIQUENAMELENGTHMAX = 64)'): 'Level-1', 'Level-2', 'Level-3', 'Level-4', 'Class', 'Kind', and 'Case'.
- The FTP directory structure has root `<FTP-ROOT>` = `/services/ftp/pub/Gaia/data/CalParam/` = `/usr4/users/gaialib/pubftp/data/CalParam/`, as set by the declaration 'FTPDIRECTORYROOT = '/usr4/users/gaialib/pubftp/data/CalParam/' '.
- Filenames used for multi-dimensional parameters and FITS-file creation have been hardcoded in the source code, but may be changed in principle (e.g., 'fits/gaiaparam_db_tables_comment.txt', 'fits/gaiaparam_db_tables_exceptions.txt', 'fits/gaiaparam_db_tables_multi.src', 'fits/gaiaparam_db_tables_directories.src', 'fits/<MULTIFILENAME>_head1.txt', 'fits/<MULTIFILENAME>_head2.txt', 'fits/<MULTIFILENAME>_head.txt', 'fits/<MULTIFILENAME>_coldes.txt', and 'fits/<MULTIFILENAME>.txt').

10 Testing updates

After changing the `<HOME>/paramdb/contents/gaiaparam_db_tables.txt` contents file, and generating the upload file `<HOME>/gaiaparam_db_tables.dat`, it is advised to first test the changes using an independent, off-line (development) implementation of the database, which is available for this purpose. This development implementation is, by construction, identical to the implementation of the nominal, on-line database. For the Gaia set-up, the database which is updated when executing a so-called test update runs on the machine `bails.estec.esa.int` and is normally refered to as the test (version of the) database. This version of the database is normally accessible using a web browser from within the ESA firewall but is not publicly accessible.

Although following the test procedure is not strictly mandatory, the advantage is that *if* the upload file does contain errors, and the update fails, the on-line version of the database will remain accessible such that availability interruptions to users are minimised. The error(s) in the contents (and upload) file(s) can thus be investigated and corrected without time pressure, after which a further test update should be performed for confirmation that the second update was successful. The final step would then be to update the nominal database as well (Section 11). If a test update is performed successfully, the nominal update — by definition — will also be successful. Note that the test procedure as described below does correctly verify multi-dimensional parameters in the contents file but does not validate FITS-file-creation aspects.

In order to perform a test update, proceed as follows:

1. Go to the correct directory: `cd <HOME>/paramdb/contents/`;
2. Update the contents file `gaiaparam_db_tables.txt` (Sections 4–8): `<EDITOR> gaiaparam_db_tables-.txt`;
3. Generate the test upload file `gaiaparam_db_tables_test.dat` by execution of the Fortran code (Section 9): `gaiaparam_db_tables.e gaiaparam_db_tables_test.dat`;

4. Update the test database: `initparamdb gaiaparam_db_tables_test.dat` (this script is discussed in the document describing implementation details behind the database, which is in preparation). Upon successful completion of the update, the message 'Parameter DB successfully initalized ...' will be displayed, which completes the test. If this message is not displayed, the update failed, requiring a correction of the contents file. In that case, go to step 2 and debug the contents file.

11 Updating the database

Before updating the on-line version of the database, it it advised to test all changes by means of a test update (refer to Section 10 for details). After successfully completing such a test update, an update of the on-line version of the database may be invoked (Figure 1 provides a schematic flow of the steps described below):

1. Go to the correct directory: `cd <HOME>/paramdb/contents/`;
2. Update the contents file `gaiaparam_db_tables.txt` (Sections 4–8): `<EDITOR> gaiaparam_db_tables-.txt`;
3. Make sure that the necessary `<MULTIFILENAME>.txt`, `<MULTIFILENAME>_coldes.txt`, and `<MULTIFILENAME>_head1.txt` files exist in `<HOME>/paramdb/contents/fits/`. Add the externally-provided FITS files to the list in `<HOME>/paramdb/contents/fits/gaiaparam_db_tables_exceptions.txt` (Section 8). This step is only relevant if changes in one or more multi-dimensional parameters were made;
4. Generate the upload file `gaiaparam_db_tables.dat` by execution of the Fortran code (Section 9): `gaiaparam_db_tables.e gaiaparam_db_tables.dat`. The `<MULTIFILENAME>_head2.txt` files for multi-dimensional parameters are created automatically;
5. If changes in one or more multi-dimensional parameters were made, proceed with step 6. Else, skip steps 6–7 and proceed with step 8 directly;
6. Verify that the FTP (sub-)directory structure for FITS files is available: `source fits/gaiaparam_db_tables_directories.src`;
7. Create all FITS files and copy them to the FTP directory (Section 8): `source fits/gaiaparam_db_tables_multi.src`;
8. Update the database: `initparamdb gaiaparam_db_tables.dat`. Upon successful completion of the update, the message 'Parameter DB successfully initalized ...' will be displayed. If this message does not appear, the advice to first ensure that the test update works without problems has been ignored and the 'eigen schuld, dikke bult'-principle applies;
9. Update the ChangeLog file, if deemed necessary, summarising changes between '<change>' and '</change>' tags: `<EDITOR> ChangeLog.xml`. Changes to parameters are not automatically accumulated/summarised in the ChangeLog file. It is the administrator's responsibility to manually maintain a list of changes at an appropriate level. It is mandatory to at least document parameter deletions and parameter-name changes in the ChangeLog file.
10. Ready.

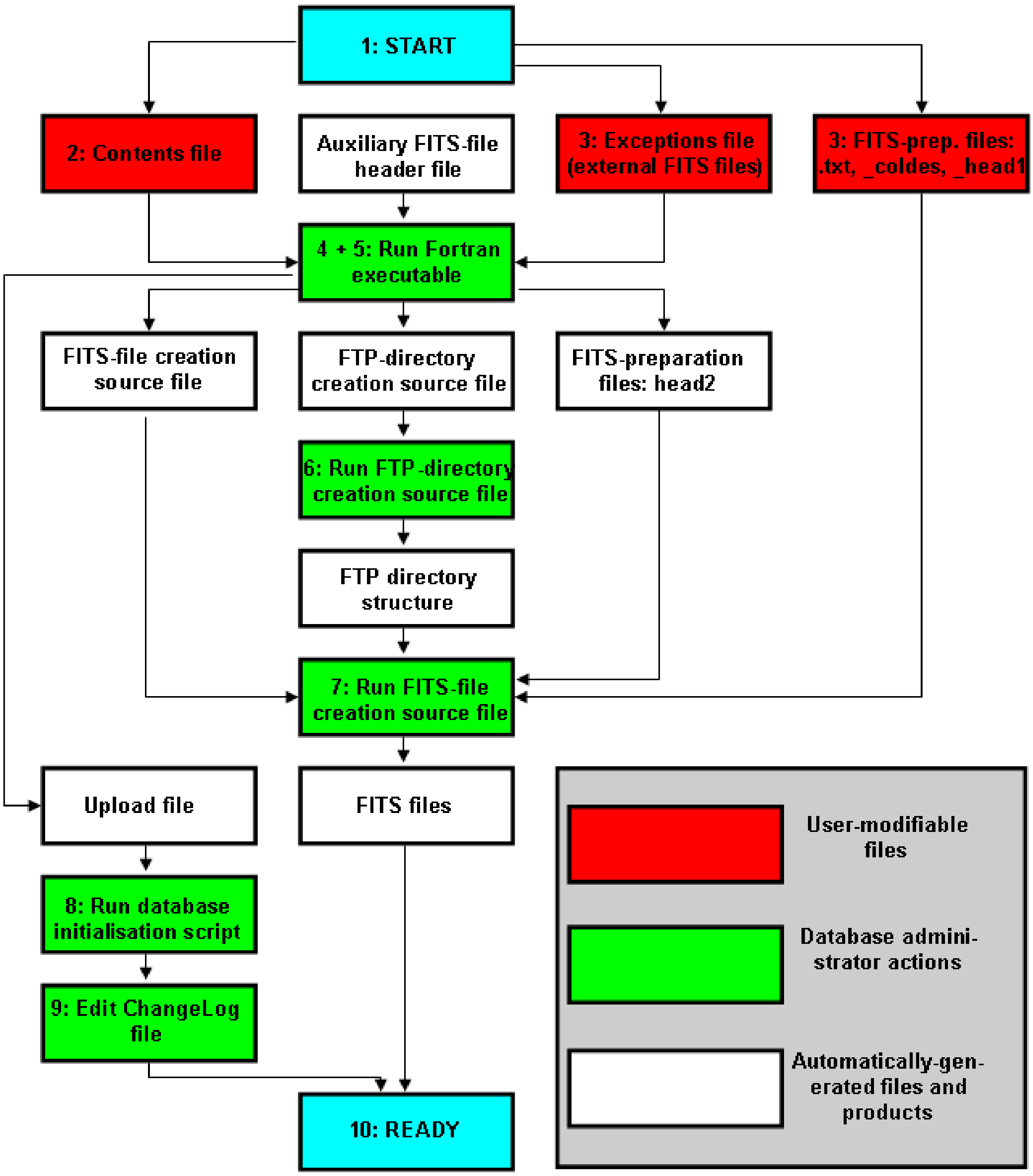
1: START
2: Contents file
Auxiliary FITS-file header file
3: Exceptions file (external FITS files)
3: FITS-prep. files: .txt, _coldes, _head1
4 + 5: Run Fortran executable
FITS-file creation source file
FTP-directory creation source file
FITS-preparation files: head2
6: Run FTP-directory creation source file
FTP directory structure
7: Run FITS-file creation source file
Upload file
FITS files
8: Run database initialisation script
9: Edit ChangeLog file
10: READY
User-modifiable files
Database administrator actions
Automatically-generated files and products

Figure 1: Schematic summary of database maintenance. Numbers refer to the steps outlined in Section 11.

12 New database releases

Database updates as described in Section 11 will always work on the current version of the database, also refered to as the 'live version' of the database. In order to create a frozen reference version of the database, which can never be changed again but which can be accessed using the 'Versions' drop-down menu on the database query page, proceed as follows:

1. Go to the correct directory: `cd <HOME>/paramdb/contents/`;
2. Make the on-line version of the database temporarily unavailable (when accessing the database through the web interface, an unavailability warning message will appear): `<HOME>/paramdb/scripts/dbwebif offline`;
3. Edit the ChangeLog file and change the version tag from '<version tag="live" ...>' to the desired version number 'X.Y': '<version tag="X.Y" ...>';
4. Actually create the new version: `makerelease -x`;
5. Introduce, at the top in the ChangeLog file, a new live-version tag like '<version tag="live" ...>';
6. Make software updates up-to-date: `cd <HOME>/paramdb/web/development/`; `mkpublic`; `cd <HOME>/paramdb/contents/`;
7. Make the cache up-to-date: `<HOME>/paramdb/scripts/cacheupdate`;
8. Make the on-line version of the database available again: `<HOME>/paramdb/scripts/dbwebif online`.

Acknowledgements

It is a pleasure to thank Karen O'Flaherty for a critical reading of the first draft of this document.

Appendix A: Examples of files

This Appendix provides some examples of what the various database files mentioned and discussed in this document look file. Recall that, for the Gaia set-up, one has the home directory set by `<HOME>` = `/usr4/users/gaialib/` and an FTP directory structure with root `<FTP-ROOT>` where `<FTP-ROOT>` = `/services/ftp/pub/Gaia/data/CalParam/` = `/usr4/users/gaialib/pubftp/data/CalParam/`.

Example contents file (`gaiaparam_db_tables.txt`)

The following shows an example of how the primary contents file, where all administrator changes to the contents of the database shall be made, shall look like:

```
# Main input file for the Gaia parameter database. Changes should (only) be made here.
#
# Do not remove this line; it is really required for initialisation!
NULL |NULL |NULL |NULL |NULL |NULL |NULL |NULL |NULL |  | ''| ''| ''| ''| ''| ''| ''| ''
   | ''| ''| '2006-03-01T09:00:00'
#
Nature       |     |    |Pi                             |Constant
   |  |                         |CONF| 0| '3.14159265358979323846264338328'
   | '%s'   | 'F'| 'basic'  | 'scalar'| ''| 'The constant Pi (also known as Archimedes\'
   constant)'| 'Well-known mathematical constant; numerical value can be extracted,
   e.g., from Mathematica 4.0 for Solaris (Wolfram Research, Inc.) using \'N[Pi,30]\''
   | ''| '\\pi'| '2006-03-01T09:00:00'
Nature       |     |    |VelocityOfLight                |Constant
   |   |Vacuum                  |CONF|                         299792458| ''
   | '%.0f' | 'D'| 'basic'  | 'scalar'| 'm s^-1'| 'Velocity of light in vacuum
   (defining constant)'| 'P.J. Mohr, B.N. Taylor, 9 December 2003, \'CODATA Recommended
   Values of the Fundamental Physical Constants: 2002\', National Institute of
   Standards and Technology, Gaithersburg, MD 20899-8401; http://www.codata.org/ and
   http://physics.nist.gov/constants (Web Version 4.0)'| ''| 'c'| '2006-03-01T09:00:00'
Nature       |     |    |AstronomicalUnit               |
   |   |Second                  |CONF|                                 0| ''
   | '%.10f'| 'D'| 'derived'| 'scalar'| 's'| 'Astronomical unit light time (TCB; SI s)'
   | 'E.M. Standish, 26 August 1998, \'JPL Planetary and Lunar Ephemerides, DE405/LE405
   \', JPL IOM 312.F-98-048; http://ssd.jpl.nasa.gov/iau-comm4/. Note that the original
   DE405 value (499.0047838061 s) is a TDB-induced value. In order to convert this
   value to a TCB-induced value, the original number should be divided by (1 -
   %LSubB_Constant%). Note that the TCB-induced numerical value 499.00478639 s listed
   in T. Fukushima, 2002, \'Report on Astronomical Constants\', Highlights of Astronomy,
   Volume 12, page 107 (Table 3: \'IAU 2000 File of Current Best Estimates\') is
   incorrect' |'499.0047838061 / (1 - %LSubB_Constant%)'| '\\tau_A'|
   '2006-03-01T09:00:00'
Nature       |     |    |AstronomicalUnit               |
   |   |Meter                   |CONF|0| ''| '%.12e'| 'D'| 'derived'| 'scalar'
   | 'm'| 'Astronomical unitlength (TCB; SI m)'| ''| '%AstronomicalUnit_Second% *
   %VelocityOfLight_Constant_Vacuum%'| 'c \\tau_A'| '2006-03-01T09:00:00'
Nature       |     |    |Parsec                         |
   |   |Meter                   |CONF|                                 0| ''
   | '%.14e'| 'D'| 'derived'| 'scalar'| 'm'| 'Parsec expressed in m'| ''|
   '%AstronomicalUnit_Meter% * 180 * 3600 / %Pi_Constant%'| ''| '2006-03-01T09:00:00'

<...>

Satellite    |SM   |    |  |CCD                              |QE
   |   |160K                          |TBC | 0|                  '%SELF%_001.fits'| '%s'
   | 'T'| 'basic'  | 'multi' | ''| 'Typical CCD quantum efficiency (QE) for T = 160 K of
   the default (broad-band) variant of CCD91-72 from e2v Technologies, i.e., standard-
   resistivity Si (100 Ohm cm), back-thinned to a nominal thickness of 16E-6 m, basic
   surface-passivation process, and a broad-band anti-reflection coating centred on 650
   nm. This CCD type is used in SM. First column: wavelength \\lambda (in nm; from 200.0
   to 1100.0). Second column: typical QE, at T = 160 K. The typical QE is derived using
   the following four steps: 1: start with all 20 EODM devices; 2: remove the two
   devices with reduced nominal thickness (4224-01-02 and 4224-18-02); 3: reject those
   devices which do not meet the combined QE-plus-MTF e2v acceptance criteria
   (3442-17-02, 3442-22-02, 3442-11-01, 3442-23-01, and 3483-18-02); 4: calculate the
   mean, measured QE of the remaining (13) devices (and extrapolate below 400 nm and
   beyond 900 nm)'| 'EADS-Astrium, October 2005, \'Gaia: Proposal for the Implementation
   Phase\', Volume II \'Technical Proposal\', Part 1 \'Proposed Spacecraft Design and
   Development Approach\', Chapter 7 \'Payload Module Design\', Section 7.2.3.3
   \'Quantum Efficiency (QE) and Modulation Transfer Function (MTF) of the Different
   AF-CCD Variants\', page 18. Reference document: A. Short, J.H.J. de Bruijne, 26
   September 2003, \'CCD QE and MTF\', GAIA-AS-002'| ''| 'QE(\\lambda)'|
   '2006-03-01T09:00:00'

<...>
```

The last example, refering to a multi-dimensional parameter, is consistently taken as example later in this Appendix.

Example upload file (`gaiaparam_db_tables.dat`)

The following shows an example of how the upload file, which is automatically generated from the contents file using the Fortran executable, shall look like:

```
use gaiaparam;

drop table if exists STATUS;
create table STATUS
(
        id smallint not null,
        primary key (id),
        name varchar(12)
);
insert into STATUS values (   0, NULL);
insert into STATUS values (   1,'CONF');
insert into STATUS values (   2,'TBC');

drop table if exists CASES;
create table CASES
(
        id smallint not null,
        primary key (id),
        name char(64)
);
insert into CASES values (   0, NULL);
insert into CASES values (   1,'Nu');
insert into CASES values (   2,'160K');
insert into CASES values (   3,'JohnsonCousinsB');
insert into CASES values (   4,'JohnsonCousinsV');
insert into CASES values (   5,'JohnsonCousinsR');
insert into CASES values (   6,'JohnsonCousinsI');
insert into CASES values (   7,'NumberOfPhotons');
insert into CASES values (   8,'NumberOfPhotonsHighResolution');
insert into CASES values (   9,'Lambda');
insert into CASES values (  10,'Vacuum');
insert into CASES values (  11,'Meter');
insert into CASES values (  12,'AstronomicalUnit');

<...>

drop table if exists PARAM_REAL;
create table PARAM_REAL
(
        id int not null,
        primary key (id),
        scope0_id smallint,
        scope1_id smallint,
        scope2_id smallint,
        scope3_id smallint,
        class_id smallint,
        kind_id smallint,
        direction_id tinyint,
        case_id smallint,
        status_id smallint,
        value double not null,
        value_s longtext,
        format varchar(64),
        value_type char(1) not null,
        value_attrib0 enum("basic", "derived"),
        value_attrib1 enum("scalar", "multi"),
        unit varchar(64),
        description longtext,
        source longtext,
        expression longtext,
        notation varchar(255),
        lastmoddate timestamp(14)
);

insert into PARAM_REAL values (  43,   1,   0,   0,   0,  32,  12,   0,   0,   1,
0, '3.14159265358979323846264338328', '%s'   , 'F', 'basic'  , 'scalar', '', 'The
constant Pi (also known as Archimedes\' constant)', 'Well-known mathematical constant;
numerical value can be extracted, e.g., from Mathematica 4.0 for Solaris (Wolfram
Research, Inc.) using \'N[Pi,30]\'', '', '\\pi', '2006-03-01T09:00:00');

insert into PARAM_REAL values (  45,   1,   0,   0,   0,  34,  12,   0,  10,   1,
299792458, '', '%.0f' , 'D', 'basic'  , 'scalar', 'm s^-1', 'Velocity of light in
vacuum (defining constant)', 'P.J. Mohr, B.N. Taylor, 9 December 2003, \'CODATA
Recommended Values of the Fundamental Physical Constants: 2002\', National Institute of
Standards and Technology, Gaithersburg, MD 20899-8401; http://www.codata.org/ and
http://physics.nist.gov/constants (Web Version 4.0)', '', 'c', '2006-03-01T09:00:00');

insert into PARAM_REAL values (  70,   1,   0,   0,   0,  50,   0,   0,  18,   1,
0, '', '%.10f', 'D', 'derived', 'scalar', 's', 'Astronomical unit light time (TCB; SI
s)', 'E.M. Standish, 26 August 1998, \'JPL Planetary and Lunar Ephemerides, DE405/
LE405\', JPL IOM 312.F-98-048; http://ssd.jpl.nasa.gov/iau-comm4/. Note that the
original DE405 value (499.0047838061 s) is a TDB-induced value. In order to convert
this value to a TCB-induced value, the original number should be divided by (1 -
%LSubB_Constant%). Note that the TCB-induced numerical value 499.00478639 s listed in
T. Fukushima, 2002, \'Report on Astronomical Constants\', Highlights of Astronomy,
Volume 12, page 107 (Table 3: \'IAU 2000 File of Current Best Estimates\') is incorrect
(typo confirmed by T. Fukushima, priv. comm., 7 February 2004)' ,'499.0047838061 / (1 -
%LSubB_Constant%)', '\\tau_A', '2006-03-01T09:00:00');

<...>
```

Example auxiliary FITS-file header file (`fits/gaiaparam_db_tables_comment.txt`)

The following shows an example of how the auxiliary FITS-file header file, used in the updating of FITS-file headers of externally-provided FITS files (see footnotes 2 and 3), shall look like:

```
- ORIGIN
- TELESCOP
- INSTRUME
- COMMENT
- COMMENT
- COMMENT
- COMMENT
- COMMENT
- COMMENT
- COMMENT
- COMMENT
- COMMENT
- COMMENT
- COMMENT
- COMMENT
- COMMENT
- COMMENT
- COMMENT
- COMMENT
- COMMENT
- COMMENT
- COMMENT
- COMMENT
<...>
```

This file, which removes selected, existing keywords in FITS-file headers of externally-provided FITS files (which are to be replaced by database-generated keywords from the `_head.txt` file; see below), shall normally not be changed by the database administrator.

Example FITS exceptions file (`fits/gaiaparam_db_tables_exceptions.txt`)

The following shows the full listing of the FITS exceptions file (see footnote 3) used in the Gaia set-up:

```
Nature_Sky_ObjectDensity_001
Nature_Earth_Ephemeris_002
Nature_Jupiter_Ephemeris_002
Nature_Mars_Ephemeris_002
Nature_Neptune_Ephemeris_002
Nature_Saturn_Ephemeris_002
Nature_Sun_Ephemeris_002
Nature_Uranus_Ephemeris_002
Nature_Venus_Ephemeris_002
```

The above FITS files have been delivered by external parties and only the headers (to be more precise: only selected header keywords) of these files are modified by execution of the primary script-like source file `fits/gaiaparam_db_tables_multi.src`. The database administrator shall manually maintain this list, including the running (version) number of the FITS files.

Example FTP-directory-creation file (`fits/gaiaparam_db_tables_directories.src`)

The following shows an example of how the FTP-directory-creation file shall look like:

```
<...>
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Nature/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Nature/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Nature/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/SM/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/AF/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/SM/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/AF/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/BP/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/RP/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/RVS/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/RVS/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/SM/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/AF/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/SM/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/AF/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/BP/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/RP/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/RVS/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/RVS/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/
mkdir -p /usr4/users/gaialib/pubftp/data/CalParam/Satellite/
<...>
```

This file is created automatically by the Fortran executable from the contents file and shall not be edited by the database administrator. The file shall be executed though by the administrator each time multi-dimensional parameters are ‘maintained’ (updated, added, ...). See Section 11 for details.

Example FITS-creation source file (`fits/gaiaparam_db_tables_multi.src`)

The following shows an example of how the FITS-creation source file shall look like:

```
\rm Nature_Sky_ObjectDensity_head.txt
cat Nature_Sky_ObjectDensity_head2.txt > Nature_Sky_ObjectDensity_head.txt
fthedit 'Nature_Sky_ObjectDensity_001.fits[0]' @gaiaparam_db_tables_comment.txt
fthedit 'Nature_Sky_ObjectDensity_001.fits[0]' @Nature_Sky_ObjectDensity_head.txt
fparkey 'Gaia/ESTEC' 'Nature_Sky_ObjectDensity_001.fits[0]' ORIGIN add=yes insert=EXTEND
fparkey 'Gaia' 'Nature_Sky_ObjectDensity_001.fits[0]' TELESCOP add=yes insert=ORIGIN
fparkey 'UNKNOWN' 'Nature_Sky_ObjectDensity_001.fits[0]' INSTRUME add=yes
        insert=TELESCOP
scp Nature_Sky_ObjectDensity_001.fits gaialib@rssd-svr:/usr4/users/gaialib/pubftp/data/
        CalParam/Nature/Sky_ObjectDensity_001.fits

<...>

\rm Satellite_SM_CCD_QE_160K_head.txt
cat Satellite_SM_CCD_QE_160K_head1.txt Satellite_SM_CCD_QE_160K_head2.txt >
        Satellite_SM_CCD_QE_160K_head.txt
fcreate 'Satellite_SM_CCD_QE_160K_coldes.txt' 'Satellite_SM_CCD_QE_160K.txt'
        !'Satellite_SM_CCD_QE_160K_001.fits' headfile='Satellite_SM_CCD_QE_160K_head.txt'
fparkey 'Gaia/ESTEC' 'Satellite_SM_CCD_QE_160K_001.fits[0]' ORIGIN add=yes insert=EXTEND
fparkey 'Gaia' 'Satellite_SM_CCD_QE_160K_001.fits[0]' TELESCOP add=yes insert=ORIGIN
fparkey 'UNKNOWN' 'Satellite_SM_CCD_QE_160K_001.fits[0]' INSTRUME add=yes
        insert=TELESCOP
scp Satellite_SM_CCD_QE_160K_001.fits gaialib@rssd-svr:/usr4/users/gaialib/pubftp/data/
        CalParam/Satellite/SM/CCD_QE_160K_001.fits

<...>
```

This file is created automatically by the Fortran executable from the contents file and shall not be edited by the database administrator. The file shall be executed though by the administrator each time multi-dimensional parameters are 'maintained' (updated, added, ...). See Section 11 for details. Note the difference between the two examples: the first example refers to an externally-provided FITS file and the `fcreate` command to actually create the FITS file, for example, is therefore missing.

Example FITS ASCII-data file (`fits/Satellite_SM_CCD_QE_160K.txt`)

The following shows an example of how FITS ASCII-data files, defining the multi-dimensional data to be included in the FITS files, shall look like:

```
200.0 0.0
250.0 0.0
275.0 0.0
300.0 10.0
350.0 34.0
400.0 57.2
450.0 74.8
500.0 86.4
550.0 93.4
600.0 92.9
650.0 90.6
700.0 82.4
750.0 73.1
800.0 59.4
850.0 44.0
900.0 28.0
950.0 16.0
1000.0 4.0
1025.0 0.0
1050.0 0.0
1100.0 0.0
```

In this example, the ASCII file has two columns, one for wavelength and one for QE. This file shall be created by the database administrator.

Example FITS-column description file (`fits/Satellite_SM_CCD_QE_160K_coldes.txt`)

The following shows an example of how the FITS-column description file, defining the FITS-column names and FITS-data properties, shall look like:

```
Wavelength D nm
QE D
```

This file shall be created by the database administrator. Each FITS column (wavelength and QE in this example) has its own line, defining (a) column name (enclose the name within double quotes for column names with internal spaces/blanks), (b) data descriptor (D for double precision), and (c) unit (optional).

Example FITS-header file (`fits/Satellite_SM_CCD_QE_160K_head.txt`)

The following shows an example of how the FITS-header file shall look like:

```
TDISP1='F5.1    '
TDISP2='F5.4    '
COMMENT Description:
COMMENT Typical CCD quantum efficiency (QE) for T = 160 K of the default
COMMENT (broad-band) variant of CCD91-72 from e2v Technologies, i.e.,
COMMENT standard-resistivity Si (100 Ohm cm), back-thinned to a nominal
COMMENT thickness of 16E-6 m, basic surface-passivation process, and a
COMMENT broad-band anti-reflection coating centred on 650 nm. This CCD type is
COMMENT used in SM. First column: wavelength \\lambda (in nm; from 200.0 to
COMMENT 1100.0). Second column: typical QE, at T = 160 K. The typical QE is
COMMENT derived using the following four steps: 1: start with all 20 EODM
COMMENT devices; 2: remove the two devices with reduced nominal thickness
COMMENT (4224-01-02 and 4224-18-02); 3: reject those devices which do not meet
COMMENT the combined QE-plus-MTF e2v acceptance criteria (3442-17-02,
COMMENT 3442-22-02, 3442-11-01, 3442-23-01, and 3483-18-02); 4: calculate the
COMMENT mean, measured QE of the remaining (13) devices (and extrapolate below
COMMENT 400 nm and beyond 900 nm)
COMMENT
COMMENT Source:
COMMENT EADS-Astrium, October 2005, \'Gaia: Proposal for the Implementation
COMMENT Phase\', Volume II \'Technical Proposal\', Part 1 \'Proposed
COMMENT Spacecraft Design and Development Approach\', Chapter 7 \'Payload
COMMENT Module Design\', Section 7.2.3.3 \'Quantum Efficiency (QE) and
COMMENT Modulation Transfer Function (MTF) of the Different AF-CCD Variants\',
COMMENT page 18. Reference document: A. Short, J.H.J. de Bruijne, 26 September
COMMENT 2003, \'CCD QE and MTF\', GAIA-AS-002
COMMENT
COMMENT Expression:
COMMENT
COMMENT LaTeX label:
COMMENT QE(\\lambda)
COMMENT
COMMENT Last modified:
COMMENT 2006-03-01T09:00:00
```

This file is created automatically by the Fortran executable from the contents file and shall not be edited by the database administrator. In fact, this header file is the straightforward concatenation of the header-1 and header-2 files (see next).

Example FITS-header-1 file (`fits/Satellite_SM_CCD_QE_160K_head1.txt`)

The following shows an example of how the FITS-header-1 file shall look like:

```
TDISP1='F5.1     '
TDISP2='F5.4     '
```

This file shall be created by the database administrator. Each FITS column (wavelength and QE in this example) has its own line, defining the display format.

Example FITS-header-2 file (`fits/Satellite_SM_CCD_QE_160K_head2.txt`)

The following shows an example of how the FITS-header-2 file shall look like:

```
COMMENT Description:
COMMENT Typical CCD quantum efficiency (QE) for T = 160 K of the default
COMMENT (broad-band) variant of CCD91-72 from e2v Technologies, i.e.,
COMMENT standard-resistivity Si (100 Ohm cm), back-thinned to a nominal
COMMENT thickness of 16E-6 m, basic surface-passivation process, and a
COMMENT broad-band anti-reflection coating centred on 650 nm. This CCD type is
COMMENT used in SM. First column: wavelength \\lambda (in nm; from 200.0 to
COMMENT 1100.0). Second column: typical QE, at T = 160 K. The typical QE is
COMMENT derived using the following four steps: 1: start with all 20 EODM
COMMENT devices; 2: remove the two devices with reduced nominal thickness
COMMENT (4224-01-02 and 4224-18-02); 3: reject those devices which do not meet
COMMENT the combined QE-plus-MTF e2v acceptance criteria (3442-17-02,
COMMENT 3442-22-02, 3442-11-01, 3442-23-01, and 3483-18-02); 4: calculate the
COMMENT mean, measured QE of the remaining (13) devices (and extrapolate below
COMMENT 400 nm and beyond 900 nm)
COMMENT
COMMENT Source:
COMMENT EADS-Astrium, October 2005, \'Gaia: Proposal for the Implementation
COMMENT Phase\', Volume II \'Technical Proposal\', Part 1 \'Proposed
COMMENT Spacecraft Design and Development Approach\', Chapter 7 \'Payload
COMMENT Module Design\', Section 7.2.3.3 \'Quantum Efficiency (QE) and
COMMENT Modulation Transfer Function (MTF) of the Different AF-CCD Variants\',
COMMENT page 18. Reference document: A. Short, J.H.J. de Bruijne, 26 September
COMMENT 2003, \'CCD QE and MTF\', GAIA-AS-002
COMMENT
COMMENT Expression:
COMMENT
COMMENT LaTeX label:
COMMENT QE(\\lambda)
COMMENT
COMMENT Last modified:
COMMENT 2006-03-01T09:00:00
```

This file is created automatically by the Fortran executable from the contents file and shall not be edited by the database administrator.

Example ChangeLog file (`ChangeLog.xml`)

The following shows an example of how the ChangeLog file shall look like:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<GaiaParamDBChangeLog>

<version tag="live" author="JdB/UL" date="">
    <change>Removed polynomial approximate relation :Satellite:RVS:
            Magnitude_VMinC1M861 and introduced :Satellite:RVS:Magnitude_
            GMinC1M861</change>
    <change>Updated polynomial approximate relation :Satellite:Magnitude_
            VMinG</change>
    <change>Updated :Satellite:VideoChain_DynamicRange (to 240 ke-)</change>
    <change>Renamed :Satellite:FoV to :Satellite:FoV_Order</change>
    <change>Removed duplicate entry :Satellite:Telescope_Number</change>
    <change>Corrected :Satellite:WFS:CCD_Position_NominalY</change>
    <change>Added :Satellite:FoV_Orientation_AC</change>
</version>

<version tag="2.0" author="JdB/UL" date="2006-03-01">
    <change>Major overhaul of contents to be in line with EADS-Astrium
            Gaia-3 design</change>
</version>

<version tag="1.2" author="JdB/UL" date="2005-07-26">
    <change>Introduced new notation for sampling and windowing strategy</change>

    <...>

    <change>Removed parameters :Nature:L2_OrbitalAmplitudeX/Y/Z_Minimum</change>
    <change>Processed GaiaSYS.NT.00137.T.ASTR (issue 1, revision 0)</change>
    <change>Changed Level-2 from CDMS to PDHS (SSMM and DataCompression
            parameters)</change>
    <change>15-deg L2 Lissajous orbit (propulsion budget and orbital
            amplitude)</change>
</version>

<version tag="1.0" author="JdB/UL" date="2004-03-01">
    <change>First full version after extensive contents review</change>
</version>

</GaiaParamDBChangeLog>
```

This file shall be manually maintained, at an appropriate level, by the database administrator (changes to parameters are not automatically accumulated/summarised in the ChangeLog file). It is mandatory to at least document parameter deletions and parameter-name changes in the ChangeLog file.