The Gaia Parameter Database

Technical User Manual

GAIA-UL-001
Revision 1.3

Uwe Lammers
ulammers@rssd.esa.int

April 27, 2004

Revision history

Rev. no.	Date	Author	Comments
0.1	2003-08-14	UL	first draft
1.0	2003-08-15	UL	JdB comments incorporated
1.1	2004-03-17	UL	updates after implementation overhaul and completed DB contents review
1.2	2004-03-26	UL	procedure for mirroring multi-dimensional datasets modified
1.3	2004-04-27	UL	minor corrections and changes after JdB comments

Contents

1 Introduction

1.1 Purpose and Scope

The Gaia Parameter Database is a central repository of parameter data pertaining to the various technical and scientific aspects of the mission. It is intended to be used as the primary source for complete, consistent, and up-to-date mission parameters by everybody involved in the design, implementation, and/or scientific support of the project.

The system is hosted in the Research and Scientific Support Department of ESA and accessible via the World-Wide Web by means of standard HTML Web browsers and related tools. The present brief document provides all necessary information required to use the system in the intended way. For information on conceptual and design issues please consult [1].

1.2 Intended readership and outline

The document is aimed at all users of the database as a technical reference and usage guide. The system has been designed as a Web-based database application and as such is meant to be self-explanatory as much as possible. There are however various usage aspects which may not be immediately obvious, especially to a first-time user. In addition to the online facility (Sect. 3.1) the system provides an offline mode whose usage requires information which is only available here (Sect. 3.2). Rendering of the database contents to various output formats including a selected number of programming languages, LaTeX and PDF is supported. Information on the structure and utilisation of the generated code is given in Sect. 4.3–4.10.

Finally, the interested reader shall find in Appendix A some details on employed methodologies and implementation techniques. This information is not needed for a mere user but should be consulted by everybody who wishes to gain a better understanding of the system's capabilities and limitations.

1.3 Support

The database is actively maintained at the Research and Scientific Support Department of ESA with support from the Gaia user community. This activity is aimed to ensure that the system reflects the most up-to-date and scientifically accurate status of relevant Gaia parameters at all times. All inquiries and questions pertaining to the contents, usage, or any other aspect of the system should be directed to

`gaialib@rssd.esa.int`

2 Version control

A large number of parameters in the database cannot be regarded as static but are expected to evolve with time as the project progresses into subsequent phases and the satellite and payload design matures. The frequency of updates cannot be predicted but is likely to be high

and also time variable. Stability of parameter values over timescales of typically months is however required for software-based simulations, optimisation studies, etc. To facilitate database usage in these cases a version control scheme is in place: Controlled revisions of the entire contents are made at irregular intervals as deemed necessary. These are labeled with a tag of the form V$major$-$minor$. $minor$ is an integer number $>= 0$ that gets incremented by one with every new revision or reset to 0 when the major revision number (positive integer) is incremented by one. A possible sequence of database revisions is, e.g. V1-0, V1-1, V1-2, V2-0, V3-0, V3-1, etc.

Both the online and offline access methods allow the selection of all available revisions in addition to the most recent one which is called the *live* version. A newly made release is identical to the *live* version until a parameter is updated. All major changes made to the *live* version are logged and visible at all times. At the time of a new release all the entries accumulated since the last release make up the change-log of the new release.

3 Usage

The system has been designed as a Web-based database application with its central entry point at

`http://www.rssd.esa.int/Gaia/paramdb`

Under this base URL two different access methods are supported, viz.

- online
- offline

Both are described in detail in the below sections 3.1 and 3.2 respectively. For any access from outside the `rssd.esa.int` domain authentification with a valid username and password is required. Please contact the database librarian (Sect. 1.3) to inquire this data.

3.1 Online usage

Online usage refers to accessing and interacting with the database through a Web browser. Until now the system has only been tested with `Netscape/Mozilla` and Microsoft's `Internet Explorer` it is however believed that other standard HTML-compliant browsers can likewise be utilised. The above URL leads to a central page that is subdivided into three main parts:

1. Header area:
   It contains a title and a convenience link for providing feedback (problems, change requests, etc.) to the database librarian (see Sect. 1.3). When followed the link brings up an email composer window with the addressee correctly set.
2. Navigation bar:
   This is a narrow, vertically oriented pane on the left hand side as shown in Fig. 1.

Under the three headings *Versions*, *Access*, and *Documentation* it provides distinct links which, when activated/followed influence the contents of the central display area.

- Versions:
  A drop-down menu shows the list of available database content versions with their respective tags as detailed in Sect. 2. By default the *live* version is selected and the central display area shows the database query form (see Sect. 3.1.1). Choosing any version other than *live* will cause the form to disappear and get replaced with a menu that allows to select an output format. This reflects the fact that it is only possible to search in the *live* version — all other database versions can merely be retrieved in their entirety.

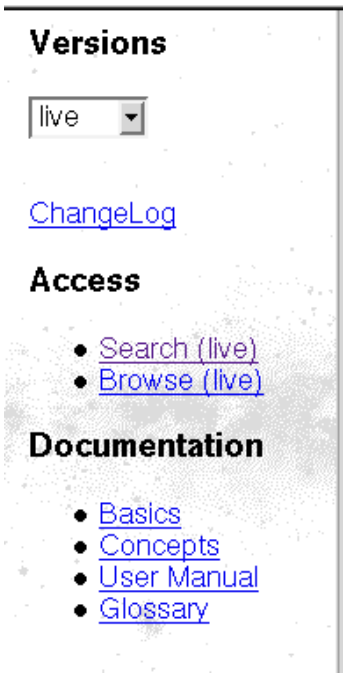

Figure 1: Navigation bar with version selector and hyperlinks

  Following the `ChangeLog` hyperlink below the version selector will display the database change history with version tags and release dates.
- Access:
  Two distinct access interfaces to the database contents are offered, viz. *Search* and *Browse*. They are initiated through following the corresponding hyperlinks in the navigation bar under *Access*. Clicking on *Search* will cause a database query form to appear in the central display area (see below Sect. 3.1.1) while *Browse* will display the contents in a graphical tree form with expandable and collapsible nodes (see Sect. 3.1.2).
  Usage of both interfaces is limited to the live version of the database only. If a version different from *live* has been selected (see above) and subsequently either *Search* or *Browse* is clicked the corresponding interface will appear in the central display area and access the live version of the database.
- Documentation:
  Clicking on either of the links offered under this heading will display sup-

porting documentation in the central area. There is some basic introductory text, a glossary of terms, and links to an overview/concepts documents as well as this present one.

3. Central display area:
   This is the large central area right to the navigation bar. It's contents varies depending on user choices made in the navigation bar and is also used to display database search results (see Sect. 3.1.1).

It should be noted that the vertical separator line between the navigation bar and the central display area is movable.

3.1.1 Query form

The query search form is shown in Fig. 2. Its purpose it to define criteria for selecting and displaying a distinct subset of the database contents. It consists of a number of entry fields:

field name	type	allowed values	name in GET-request	comment
Level-1	choice menu	indicated	`choiceFields[0]`	top-level identifiers
Level-2	choice menu	indicated	`choiceFields[1]`	level-2 identifiers
Level-3	choice menu	indicated	`choiceFields[2]`	level-3 identifiers
Level-4	choice menu	indicated	`choiceFields[3]`	level-4 identifiers
Class	free-text field	wildcard string	`classField`	wildcard match of parameter class name
Kind	choice menu	indicated	`choiceFields[5]`	valid kind identifiers
Direction	choice menu	indicated	`choiceFields[6]`	valid direction identifiers
Case	choice menu	indicated	`choiceFields[7]`	valid case identifiers
Status	choice menu	indicated	`choiceFields[8]`	valid status identifier
Basic flag	radio button	`all` | `basic` | `derived`	`basicFlag`	`all` selects basic and derived parameters
Scalar flag	radio button	`all` | `scalar` | `multidimensional`	`scalarFlag`	`all` selects scalar and multidimensional parameters
Description	free-text field	wildcard string	`descField`	wildcard match of parameter description field
Source	free-text field	wildcard string	`srcField`	wildcard match of parameter description field
Render target	choice menu	indicated	`renderTarget`	selects output format

Table 1: Available database search from entry fields. For the meaning of the fourth column please refer to Sect. 3.2

Each of these fields (except the last one) corresponds to an existing native attribute in the database. For their exact meaning please consult [1] or push the buttons marked **?** which form a column on the left hand side of the form. They will cause a small pop-up window with explanatory text to appear.

After the form has been filled pushing the **Search** button will cause a query to the database for those parameters that match the form data. The result set will subsequently be displayed in the selected format within the Web browser window.

The usage of the query form is fairly obvious but one may want to take notice of the following:

- A filled out query form represents a single logical expression with which the database is queried. Vertically, the fields are combined through a logical *AND* and horizontally (within the scope of an attribute) the logical junction is *OR*.
- The number of *OR* operands within a row is unlimited - pushing the **More** button will show a new choice field with a default `-Any-`. If more than one *OR* operand is visible a **Less** button will be generated with which the respective right-most operand can be deleted. The **Less** button will disappear again when there is only one value left.
- A displayed value `-Any-` indicates that the field value is indifferent, i.e., all field values in the database will match this.
- The initial state of the form corresponds to a selection expression which selects the *entire* database contents. So, pushing **Search** on the unaltered form will render the complete database into the selected output format.
- The three fields `Class`, `Description`, and `Source` offer simple regular expression matching with the two supported wild-card characters:

  `*` : matches zero-or more characters
  `?` : matches a single character

  All other characters match themselves in a case-insensitive manner. Example: The Class value `Napier` is matched by the expressions `*`, `napier`, `*API*`, `*Napie?`, etc. Note that the expression `*Napier` matches any string that ends with the word `Napier` and does not continue, i.e., `NapierConstant` is *not* matched.
- The form state is persistent. All selections are remembered across searches and even different sessions. To clear the state and re-initialize the form push the **Reset** button.
- The fields *basic-* and *scalar flag* are represented by three-choice radio buttons. The default value `all` stands for *indifferent*, i.e., the corresponding attribute can have either of the two other values.
- The values shown in the drop-down menus of the choice fields are not static entries but are dynamically generated through a transparent initial query to the database when the form is being accessed for the first time (per session). This means that they necessarily present valid values of the respective parameter attributes at all times.
- In spite of the preceding statement the form does not prohibit the generation of illogical queries which lead to empty result sets (e.g.,

  `choiceFields[0][0]=Nature&choiceFields[6][0]=AL`

  see 3.2). This was a deliberate design choice after trading-off desirable user convenience and implementation complexity.

- The form allows only the construction of database queries of the logical pattern $(a0$ `OR` $a1$ `OR`$\ldots)$ `AND` $(b0$ `OR` $b1\ldots)\ldots$ whereas of course more complex expressions are conceivable. Lacking a clear need for such functionality this is regarded as an acceptable limitation of the current design.
- Search results are shown in the central display area for all selectable output formats (see Sects.4.1–4.10). The default one is HTML aimed at and optimized for a human reader. Except for PDF, all other formats have been designed to be used in conjunction with software readers. To this end the output needs to be saved as a local disk file on the machine on which the browser is run. The browser's documentation should be consulted on how to perform this if unknown (e.g. in Netscape/Mozilla: Right mouse-click in browser frame, then select `This frame`$\longrightarrow$`Save Frame As...` and choose a suitable file name).

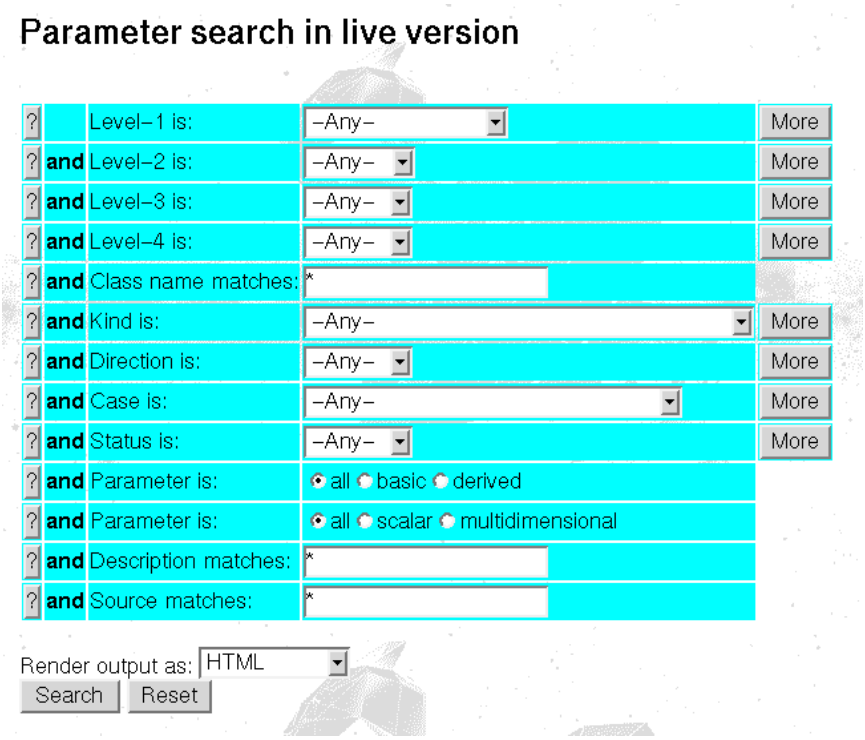

Figure 2: Query search form interface to the database. The various field values are logically combined to form a single query string for the database backend.

3.1.2 Browse interface

The browse interface provides navigation through the database parameter hierarchy using a tree-like display with collapsible and expandable nodes. A displayed basic parameter is initially presented by a clickable hyperlink only. Following the link will then show all the parameter's attributes in the same format as the HTML output (Sect. 4.1) obtained via the search form (Sect. 3.1.1). Fig. 3 shows an example of a partially expanded view on the parameter hierarchy.

Figure 3: Browse interface to the database contents. The nodes represent levels in the parameter hierarchy - expanding a node opens and shows all the parameters and sublevels below it.

3.2 Offline

Offline usage refers to accessing the database through offline network programs such as `wget`[1]. In the following the description will focus on `wget` but there are other tools which offer identical or similar functionality and can be availed as well. One such tool is for instance `curl`[2].

In Appendix A it is explained that the online interface to the database uses the `HTTP POST` method to communicate with a software layer between the browser and the database backend. This is a standard Web technique which in this case ensures the transparent propagation of the user-provided field values from the query form (Sect. 3.1.1) to the database frontend software.

There is another scheme called `HTTP GET` which is very similar to `POST` except that all the parameters are passed to the Web server as part of the URL. This is used when accessing the database offline with `wget`. The URL is formed from the fixed string

```
http://www.rssd.esa.int/Gaia/paramdb/access-point
```

[1] `http://www.gnu.org/directory/wget.html`
[2] `http://curl.haxx.se`

followed by an optional list of `key=value` pairs or simply `key`s separated by the character &. Two distinct access points are available:

1. `access-point=search.php?submit&`
   This access point provides the exact same functionality as the online query form (Sect. 3.1.1). The allowed list of `key`s is given in the fourth column of Table 1 and the associated values are given by the strings in the drop-down menus on the online query form. In addition to this the key `renderTarget` may take one of the values detailed in Table 2. Omitting this key will default to XML as render target.

   All queries made through this access point access the *live* version of the database.

2. `access-point=retrieve.php?`
   This access point is intended for the retrieval of complete database versions other than *live*. Three keys are supported:

   (a) `version=V`x`-`y
   This selects the database with the major/minor version number x/y for retrieval. The list of available versions should be inquired through an online access (Sect. 3.1).

   (b) `renderTarget=`$what$
   This selects a rendering target for the output. The list of available choices is given in Table 2. Omitting this key will default to XML as output format.

   (c) `listformats`
   This key can be used to obtain a list of all available output formats. The result is a table with three columns and one row per format option:

```
HTML .html http://www.rssd.esa.int//Gaia/paramdb/retrieve.php?renderTarget=HTML
XML .xml http://www.rssd.esa.int//Gaia/paramdb/retrieve.php?renderTarget=XML
Java .java http://www.rssd.esa.int//Gaia/paramdb/retrieve.php?renderTarget=Java
C++ .h http://www.rssd.esa.int//Gaia/paramdb/retrieve.php?renderTarget=Cxx
ANSI-C _c.h http://www.rssd.esa.int//Gaia/paramdb/retrieve.php?renderTarget=C
Fortran-90 .f90 http://www.rssd.esa.int//Gaia/paramdb/retrieve.php?renderTarget=f90
Fortran-77 .f http://www.rssd.esa.int//Gaia/paramdb/retrieve.php?renderTarget=f77
CSV .csv http://www.rssd.esa.int//Gaia/paramdb/retrieve.php?renderTarget=CSV
LaTeX .tex http://www.rssd.esa.int//Gaia/paramdb/retrieve.php?renderTarget=LaTeX
PDF .pdf http://www.rssd.esa.int//Gaia/paramdb/retrieve.php?renderTarget=PDF
```

   Col-1: Format identifier to be given as value of `renderTarget` key
   Col-2: Suggested name extension for data file - a suggested name stem is `GaiaParam`
   Col-3: Base URL to retrieve the database contents in the respective format. Note that also a version must be selected by appending `&version=V`x`-`y.

Please also take notice of the following when accessing the database offline via `wget`:

- For composing the `HTTP GET` URL the above brief explanation together with the below selection of examples shall probably suffice. The complete specification including syntax rules and character-escape mechanisms is given in the W3C RFC 1738 (`http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1738.html`).
- When used from the Unix command line the GET URL must be enclosed in single (') or double (") quotes to prevent the shell from interpreting the characters '&' and '?'.

value of `renderTarget`	action
`HTML`	arrange result set as table in HTML form (Sect. 4.1)
`XML`	show result set as XML structure (Sect. 4.2)
(empty) or missing	same as `renderTarget=XML`
`Java`	generate Java class hierarchy (Sect. 4.3)
`Cxx`	generate C++ class hierarchy (Sect. 4.4)
`ANSI-C` or `C`	generate ANSI-C source code (Sect. 4.5)
`Fortran-90` or `f90`	generate Fortran-90 source code (Sect. 4.6)
`Fortran-77` or `f77`	generate Fortran-77 source code (Sect. 4.7)
`CSV`	generate comma-separated-values ASCII output (Sect. 4.8)
`LaTeX`	generate LaTeX output (Sect. 4.9)
`PDF`	generate PDF output (Sect. 4.10)

Table 2: Possible rendering targets for database query results available online or offline for both access points `search.php?submit` and `retrieve.php?`.

- The offline query via `wget` with the `search.php&submit` access point offers the exact same functionality as online, i.e. all search parameters available in the query form are also present as corresponding GET parameters.
- Invoking `wget` with only the above core URL (so, without any other GET parameters except `renderTarget`) is equivalent to performing an online search with all the search form's default values, i.e., downloading the entire *live* version of the database.
- By default `wget` writes its output to *stdout*. The command line option `-O` can be used to write it to a file instead (see below examples).
- For accesses outside the `rssd.esa.int` domain the command line options `--http-user`/`--http-passwd` must be used to pass needed authentification data to the server
- The functionality of the **More** button on the query form in the online case is realized by suffixing the corresponding GET parameter name (column four in Table 1 by `[`n`]` where n is a running integer number starting with 0. The suffix `[0]` has always to be present even if only a single value is specified (see below examples).

3.2.1 Examples using access point `search`

In the following examples the character string `http:...` stands for
`http://www.rssd.esa.int/Gaia/paramdb/search.php?submit`

1. Retrieve the entire database contents in XML (note that a missing `renderTarget` defaults to `XML`) and write it to file `GaiaParam.xml` (authenticate as (dummy) user *astro* with password *nomer*):

   ```
   wget --http-user=astro -http-passwd=nomer -O GaiaParam.xml 'http:...'
   ```

   `--http-user`/`--http-passwd` are omitted in the below examples but authentification is needed for all accesses outside the `rssd.esa.int` domain.

2. Retrieve the entire database contents as a Java class file `GaiaParam.java`:

```
wget -O GaiaParam.java 'http:...&renderTarget=Java'
```

3. Retrieve all parameters in the top-level category `Nature` as C++ source code and write to file `GaiaNature.h`:

```
wget -O GaiaNature.h 'http:...&choiceFields[0][0]=Nature&\
renderTarget=Cxx'
```

4. Query the database for parameter `:Spectro:FocalLength-` output as Fortran-90 to file `SpecF.f90`:

```
wget -O SpecF.f90 'http:...&choiceFields[0][0]=Spectro&\
classField=FocalLength&renderTarget=Fortran-90'
```

5. Retrieve all derived along-scan parameters in either level-2 category `BBP` or `ASM` as ANSI-C source and write to file `bbp.c`:

```
wget -O bbp.c 'http:...&choiceFields[1][0]=BBP&\
choiceFields[1][1]=ASM&choiceFields[6][0]=AL&basicFlag=derived&\
renderTarget=C'
```

6. Retrieve all parameters whose class name contains the word CCD as Fortran-77 source code in file `ccd.f`:

```
wget -O ccd.f 'http:...&classField=*CCD*&renderTarget=f77'
```

3.2.2 Examples using access point `retrieve`

In the following examples the character string `http:...` stands for `http://www.rssd.esa.int/Gaia/paramdb/retrieve.php`

1. Retrieve version V1-0 of the database as a Java class file `GaiaParam.java`:

```
wget -O GaiaParam.java 'http:...?version=V1-0&renderTarget=Java'
```

2. Retrieve version V3-4 of the database rendered as a PDF document `GaiaParam.pdf`:

```
wget -O GaiaParam.java 'http:...?version=V3-4&renderTarget=PDF'
```

3. Obtain a list of all available formats:

```
wget -O GaiaParam.java 'http:...?listformats'
```

3.2.3 Automatic and repeated offline interrogation

On a Unix-based client the tool `wget` can be used in conjunction with the standard Unix *crontab* facility to download the entire database or parts of it in a fully automatic, periodic manner. This will be useful in current or future Gaia software environments such as GDAAS (and follow-up) or the simulator where the procedure could be made an integral part of an automatic and continuous integration process of these systems.

As an example, the following script `gaiadbjava` will retrieve the full database contents in Java and write the output to a file `GaiaParam.java` in the home directory:

```
#!/bin/sh
#
#   gaiadbjava: Retrieve full content of Gaia Parameter Database in Java
#
WGET=/usr/local/bin/wget
EXPORTDIR=$HOME
DBURL=http://www.rssd.esa.int/Gaia/paramdb/search.php
STEM=GaiaParam
NAMESTEM=${EXPORTDIR}/${STEM}

retrieve() {
    [ $# -ne 2 ] && echo "Usage: $0 outfile format" && exit 1

    $WGET -O "$1" "${DBURL}/?submit&renderTarget=$2"
}

#   retrieve Java class file
retrieve ${NAMESTEM}.java Java

exit 0
```

Provided the script is located in the directory `$HOME/bin` the following `crontab` entry will perform the download every night at 2am:

```
# Perform full DB retrieval every night at 2am
0 2 * * * $HOME/bin/gaiadbjava
```

For more information please consult the `crontab` manual page.

4 Rendering targets

The database contents may be retrieved in a number of different formats including HTML, XML and a few selected programming languages. The following subsections provide some more detailed information about the generated output and its intended usage.

4.1 HTML

This is the standard rendering target in the online case (Sect. 3.1). The result set is presented in a single large table composed of the columns *Name*, *Value*, *Unit*, *Basic*, *Scalar*, *Status*, *Description*, *Source*, *Expression* (formula for derived parameters, empty otherwise), *LaTeX label*, and *Last Modified* (date and time of last modification to this parameter). An example is:

Parameter search results (DB Version: –live–:2004–03–22T18:39:19)

Name	Value	Unit	Basic	Scalar	Status	Description	Source	Expression	LaTeX label	Last modified
:Nature:Pi_Constant	3.14159265358979323846264338328		true	true	CONF	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	2004–03–01T09:00:00

Back to search form

Figure 4: HTML rendering of result of database query `choiceFields[0][0]=Nature&classField=*PI*`

The **Back to search form** button provides a link back to the query form in its state prior to the query.

4.2 XML

As further explained in App. A, XML is used as an intermediate native format of the system and can thus also be selected as rendering target. This is also the default choice in the offline case (Sect. 3.2) if no format is explicitley specified.

The following XML output is generated for the database query `choiceFields[0][0]=Nature&classField=PI`:

```
<?xml version="1.0"?>
<GaiaParam version="-live-" date="2004-03-19T13:56:49">
  <Nature>
    <Parameter class="Pi" kind="Constant" direction="" case="" type="F" basic="true"
    scalar="true" status="CONF" unit="" expression="" LaTeXlabel="\pi"
    lastmoddate="2004-03-01T09:00:00">3.14159265358979323846264338328<Description>
    The constant Pi (also known as Archimedes' constant)</Description><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]'
   </Source></Parameter>
  </Nature>
</GaiaParam>
```

With XML supporting hierarchical structures the hierarchy of the parameters is directly reflected in corresponding nested XML constructs. The entire structure is embedded in the top-level element `GaiaParam` and each new parameter level is associated with a nested element of the same name (e.g. `Nature`). The parameters themselves are represented by `Parameter` elements whose text fields contain the numerical value and whose attributes are formed by the corresponding attributes of the parameter in the database. `Description` and `Source` form separate elements within the scope of the respective `Parameter` element. The date and time of the retrieval is given as attribute `date` of the top-level `GaiaParam` element in ISO

yyyy-mm-ddThh:mm:ss-format and the database version is given as a string value of the attribute version.

4.3 Java

In the case of Java output the nested parameter hierarchy is mapped onto a corresponding hierarchy of nested class definitions. The innermost classes comprise the basic scalar parameter definitions in the form

```
static public final  type  param = value;
```

statements. The parameter name is rendered in uppercase in line with common Java coding style practices. Some parameter attributes (unit etc.) are rendered as code comments and each parameter definition is preceded by a comment section that follows follows Javadoc[3] conventions. The entire class hierachy is enclosed by a top-level class GaiaParam that is part of the package gaiaparameters. It must be noted that the Java language requires the class file to reside in a directory with that same name in order to make use of in other classes. In addition this directory must appear as a component in the CLASSPATH environment variable. The constant GaiaParam.DBVersion holds a string that uniquely identifies the database version used in the query.

As an example the following Java output is generated for the database query choiceFields[0][0]=Nature&classField=PI:

```
//-----------------------------------------------------------------------------
//
//                     Gaia Science Operations Center (SOC)
//                     (c) 2003-2020 European Space Agency
//
//-----------------------------------------------------------------------------
//
//             THIS IS AN AUTOMATICALLY GENERATED FILE - DO NOT EDIT!
//
//    The file has been automatically generated from the contents of the
//    Gaia Parameter Database at the URL
//        http://www.rssd.esa.int/Gaia/paramdb
//    on 2004-03-19T14:00:24.
//
//    Please report any problems arising from the usage of this file to
//    the Gaia Librarian gaialib@rssd.esa.int
//
package gaiaparameters;

/**
 * Container class to enclose the contents of the Gaia Parameter
 * Database at<br/>
 *     <code><a href="http://www.rssd.esa.int/Gaia/paramdb">
 *     http://www.rssd.esa.int/Gaia/paramdb</a></code><p>
 * A hierachy of nested classes below matches the parameter naming scheme
 * detailed in <code><a href="ftp://ftp.rssd.esa.int/pub/Gaia/docs/gaia-jdb-007.pdf">
 * GAIA-JdB-007</a></code><br/>
```

[3] http://java.sun.com/j2se/javadoc/

```
 *
 * @author  Gaia SOC, ESA/ESTEC
 * @version -live-:2004-03-19T14:00:24
 */
final public class GaiaParam {

public static final String DBVersion = "-live-:2004-03-19T14:00:24";

final public static class Nature {
    /**
     * The constant Pi (also known as Archimedes' constant)
     * <p>
     * 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]'<br/>
     * Status: CONF<br/>
     * Basic : true<br/>
     * Scalar: true
     */
    public static final double PI_CONSTANT = 3.14159265358979323846264338328;

}

}
```

Example usage:

```
//
//  Demonstrate usage of Gaia Parameter Java export
//
import gaiaparameters.*;

public class gaiaParamJavaTest {
    public static void main(String argv[]) {
        System.out.println(GaiaParam.DBVersion);
        System.out.println(GaiaParam.Nature.PI_CONSTANT);
    }
}
```

4.4 C++

The C++ output is very similar to the Java case, i.e. the nested parameter hierarchy is mapped onto a corresponding hierarchy of nested classes and the scalar parameters are defined as static data members of the respective innermost class. The parameter attributes are rendered as code comments. All classes are part of a namespace `GaiaParam`. Because ANSI C++ does not allow the in-class initialisation of non-integer constants the string parameters are represented as static methods returning constant character pointers. The constant `GaiaParam::DBVersion` holds a string that uniquely identifies the database version used in the query.

The following C++ output is generated for the database query `choiceFields[0][0]=Nature&classField=PI`:

```
//----------------------------------------------------------------------------
```

```
//
//                     Gaia Science Operations Center (SOC)
//                     (c) 2003-2020 European Space Agency
//
//--------------------------------------------------------------------------
//
//             THIS IS AN AUTOMATICALLY GENERATED FILE - DO NOT EDIT!
//
//    The file has been automatically generated from the contents of the
//    Gaia Parameter Database at the URL
//        http://www.rssd.esa.int/Gaia/paramdb
//    on 2004-03-22T17:08:23.
//
//    Please report any problems arising from the usage of this file to
//    the Gaia Librarian gaialib@rssd.esa.int
//
#ifndef GAIA_PARAM_H
#define GAIA_PARAM_H

//
// Namespace to enclose the contents of the Gaia Parameter Database at
//     http://www.rssd.esa.int/Gaia/paramdb
// A hierachy of nested classes below matches the parameter naming scheme
// detailed in
// ftp://ftp.rssd.esa.int/pub/Gaia/docs/gaia-jdb-007.pdf
//
// Author: Gaia SOC, ESA/ESTEC
// Version: -live-:2004-03-22T17:08:23
//
namespace GaiaParam {

static const char *const DBVersion = "-live-:2004-03-22T17:08:23";

class Nature {
public:
    // The constant Pi (also known as Archimedes' constant)
    // 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]'
    // Status: CONF
    // Basic : true
    // Scalar: true
    static const double PI_CONSTANT = 3.14159265358979323846264338328;

};

}
#endif
```

Example usage:

```
//
//  Demonstrate usage of Gaia Parameter C++ export
//
#include <iostream>
#include <iomanip>

#include "GaiaParam.h"
```

```
using namespace GaiaParam;
using namespace std;

int main(void) {
    cout<<"Version="<<DBVersion<<endl;
    cout<<setprecision(20)<<Nature::PI_CONSTANT<<endl;
}
```

4.5 ANSI-C

Since there is no concept of classes and nesting of data structures in ANSI-C, the hierarchical parameter structure is converted into a flat list of static constant global scalars. The parameters names are composed of the fully qualified parameter name (path + intrinsic name - see [1]) with the ':'-separator replaced by '_'. Parameter attributes are rendered as code comments. The constant `GaiaParamDBVersion` holds a string that uniquely identifies the database version used in the query.

The following ANSI-C output is generated for the database query `choiceFields[0][0]=Nature&classField=PI`:

```
/*----------------------------------------------------------------------------
 *
 *                     Gaia Science Operations Center (SOC)
 *                     (c) 2003-2020 European Space Agency
 *
 *----------------------------------------------------------------------------
 *
 *             THIS IS AN AUTOMATICALLY GENERATED FILE - DO NOT EDIT!
 *
 *    The file has been automatically generated from the contents of the
 *    Gaia Parameter Database at the URL
 *        http://www.rssd.esa.int/Gaia/paramdb
 *    on 2004-03-19T14:28:06.
 *
 *    Please report any problems arising from the usage of this file to
 *    the Gaia Librarian gaialib@rssd.esa.int
 */
#ifndef GAIA_PARAM_H
#define GAIA_PARAM_H

typedef enum { false=0, true } bool;

static const char *const GaiaParamDBVersion = "-live-:2004-03-19T14:28:06";

/*
 * The constant Pi (also known as Archimedes' constant)
 * 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]'
 * Status: CONF
 * Basic : true
 * Scalar: true
 */
static const double Nature_PI_CONSTANT = 3.14159265358979323846264338328;
```

```
#endif
```

Example usage:

```
/*
 *  Demonstrate usage of Gaia Parameter ANSI-C export
 */
#include <stdio.h>
#include "GaiaParam.h"

int main() {
    printf("Version=%s\n", GaiaParamDBVersion);
    printf("Pi=%20.13f\n", Nature_PI_CONSTANT);

    return 0;
}
```

4.6 Fortran-90

Since there is no concept of classes and nesting of data structures in Fortran-90, the hierarchical parameter structure is converted into a flat list of parameter statements within a global module `GaiaParam`. The identifier names are composed of the fully qualified parameter name (path + intrinsic name - see [1]) with the ':'-separator replaced by '_'. Parameter attributes are rendered as code comments. Note that the length of identifiers will in most cases exceed the maximum value 31 allowed by the Fortran-90 standard. This is not deemed critical as most compilers tolerate this violation of the standard either transparently or via explicit command line options.

The module parameter `GAIAPARAM_DB_VERSION` holds a string that uniquely identifies the database version used in the query.

The following Fortran-90 output is generated for the database query `choiceFields[0][0]=Nature&classField=PI`:

```
!-------------------------------------------------------------------------------
!
!                     Gaia Science Operations Center (SOC)
!                     (c) 2003-2020 European Space Agency
!
!-------------------------------------------------------------------------------
!
!             THIS IS AN AUTOMATICALLY GENERATED FILE - DO NOT EDIT!
!
!    The file has been automatically generated from the contents of the
!    Gaia Parameter Database at the URL
!        http://www.rssd.esa.int/Gaia/paramdb
!    on 2004-03-19T14:32:14.
!
!    Please report any problems arising from the usage of this file to
!    the Gaia Librarian gaialib@rssd.esa.int
!
module GaiaParam
```

```
    integer, parameter :: DP = KIND(1D0)

    character(len=*), public,  parameter :: GAIAPARAM_DB_VERSION = &
        '-live-:2004-03-19T14:32:14'

    !
    ! The constant Pi (also known as Archimedes' constant)
    ! 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]'
    ! Status: CONF
    ! Basic : true
    ! Scalar: true
    !
    real(kind=DP), public, parameter :: NATURE_PI_CONSTANT = 3.14159265358979323846264338328

end module
```

Example usage:

```
!
!   Demonstrate usage of Gaia Parameter f90 export
!
program DbUseDemo
   use GaiaParam

   write(*, *) 'VERSION=', GAIAPARAM_DB_VERSION
   write(*, *) NATURE_PI_CONSTANT
end
```

4.7 Fortran-77

Since there is no concept of classes and nesting of data structures in Fortran-77, the hierarchical parameter structure is converted into a flat list of parameter statements. The identifier names are composed of the fully qualified parameter name (path + intrinsic name - see [1]) with the ':'-separator replaced by '_'. Parameter attributes are rendered as code comments.

The parameter `GAIAPARAM_DB_VERSION` holds a string that uniquely identifies the database version used in the query.

The following Fortran-77 output is generated for the database query `choiceFields[0][0]=Nature&classField=PI`:

```
C-------------------------------------------------------------------------------
C
C                     Gaia Science Operations Center (SOC)
C                     (c) 2003-2020 European Space Agency
C
C-------------------------------------------------------------------------------
C
C             THIS IS AN AUTOMATICALLY GENERATED FILE - DO NOT EDIT!
C
C    The file has been automatically generated from the contents of the
```

```
C    Gaia Parameter Database at the URL
C        http://www.rssd.esa.int/Gaia/paramdb
C    on 2004-03-22T18:13:39.
C
C    Please report any problems arising from the usage of this file to
C    the Gaia Librarian gaialib@rssd.esa.int
C
      IMPLICIT NONE

      CHARACTER GAIAPARAM_DB_VERSION*(*)
      PARAMETER (GAIAPARAM_DB_VERSION = '-live-:2004-03-22T18:13:39')

C
C The constant Pi (also known as Archimedes' constant)
C Source: Well-known mathematical constant; numerical value can be extracted,
C e.g., from Mathematica 4.0 for Solaris (Wolfram Research, Inc.) using 'N[Pi,30]'
C Status: CONF
C Basic : true
C Scalar: true
C
      DOUBLE PRECISION
     +  NATURE_PI_CONSTANT
      PARAMETER (
     +  NATURE_PI_CONSTANT
     +  = 3.14159265358979323846264338328d0
     +)
```

Example usage:

```
C
C   Demonstrate usage of Gaia Parameter f77 export
C
      program DbUseDemo
      include 'GaiaParam.f'

      write(*, *) 'VERSION=', GAIAPARAM_DB_VERSION
      write(*, *) NATURE_PI_CONSTANT
      end
```

4.8 CSV

CSV (Comma-separated values) is an ASCII format where the various database fields are separated by commas. The format is useful for text processing purposes and for import into other tools, e.g. Microsoft Excel. The parameters appear in their fully qualified form (path + intrinsic name - see [1]) with the ':'-separator replaced by '_'.

The following CSV output is generated for the database query `choiceFields[0][0]=Nature&classField=PI`:

```
Gaia Science Operations Center (SOC)
(c) 2003-2020 European Space Agency
```

```
THIS IS AN AUTOMATICALLY GENERATED FILE - DO NOT EDIT!

The file has been automatically generated from the contents of the
Gaia Parameter Database at the URL
    http://www.rssd.esa.int/Gaia/paramdb
Version -live-:2004-03-22T17:15:36

Please report any problems arising from the usage of this file to
the Gaia Librarian gaialib@rssd.esa.int

Name,Value,Unit,Basic,Scalar,Description,Source,Expression,LaTeX label,Last modified
Nature_Pi_Constant,3.14159265358979323846264338328,,true,true,"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",2004-03-01T09:00:00
```

4.9 LaTeX

The output in this case is a standard LaTeX file - the extra packages `a4wide`, `times`, `alltt`, `longtable`, and `portland` are used however they are normally part of the local LaTeX installation. When processed the resulting document shows the selected parameters with associated parameters typeset in landscape mode as a long multi-page table.

Please note: When producing PostScript with `dvips` the option `-t landscape` must be given to ensure the proper orientation of the table in the output. In addition, when the PostScript shall be converted into a PDF document the option `-G1` should be given to `dvips`[4]. Below is an example of how to generate PDF from LaTeX with all the required switches for `dvips` and `gs`.

The following LaTeX output is generated for the database query `choiceFields[0][0]=Nature&classField=PI`:

```
%------------------------------------------------------------------------
%
%                     Gaia Science Operations Center (SOC)
%                     (c) 2003-2020 European Space Agency
%
%------------------------------------------------------------------------
%
%             THIS IS AN AUTOMATICALLY GENERATED FILE - DO NOT EDIT!
%
%    The file has been automatically generated from the contents of the
%    Gaia Parameter Database at the URL
%        http://www.rssd.esa.int/Gaia/paramdb
%    on 2004-03-19T14:41:13.
%
%    Please report any problems arising from the usage of this file to
%    the Gaia Librarian gaialib@rssd.esa.int
%
\documentclass[10pt]{article}
\usepackage{a4wide,times,alltt,longtable,portland}
\setlength{\oddsidemargin}{0mm}
```

[4] See `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=distill-prob` for details

```
\setlength{\hoffset}{-10mm}
\setlength{\voffset}{0mm}
\setlength{\marginparwidth}{0mm}
\addtolength{\textwidth}{20mm}
\addtolength{\textheight}{20mm}
\pagestyle{headings}
\begin{document}
\landscape
\begin{center}
\Huge
Gaia Parameter Database Contents (Version: -live-:2004-03-19T14:41:13)
\end{center}

%   start of table
\setlongtables
\scriptsize
\begin{longtable}{|p{.2\textwidth}|p{.18\textwidth}|c|c|c|p{.20\textwidth}|
p{.15\textwidth}|p{.15\textwidth}|}\hline
\normalsize Name & \normalsize Value & \normalsize B & \normalsize S &
\normalsize Status & \normalsize Description &  \normalsize Source &
 \normalsize Expression\\ \hline

\multicolumn{8}{|l|}{\normalsize\bf :Nature:}\markright{\bf :Nature:}\\\hline
\tt Pi\-\_Constant & $\pi=3.14159265358979323846264338328\;\mbox{  }$ &
$\bullet$ & $\bullet$ & CONF & 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]' & \tiny\tt  \\\hline

\end{longtable}
\end{document}
```

Example usage:

```
unix> latex GaiaParam.tex
unix> dvips -G1 -t landscape GaiaParam
unix> gs -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -dOrient1=false \
    -sOutputFile=GaiaParam.pdf GaiaParam.ps
```

4.10 PDF

The output is a PDF document in the format described in Sect. 4.9. Internally, LaTeX is chosen as rendering option and the output processed and converted to PDF with the tools and command line options given above. This is all transparent to the user - since the generation of the final document takes some time the response to such a rendering request will be slower than for any of the other output formats.

5 Multi-dimensional data sets

In addition to scalar parameters the database also contains multi-dimensional data in FITS[5] format. Owing to the self-descriptive nature of FITS these datasets should not require further explanation. Ample information on structure, contents, orgin, etc. is available in the form of FITS comment cards. At the moment only single binary tables (`Bintables`) with scalar columns and single two-dimensional image extensions are used.

5.1 Naming convention

The usual naming convention employed for scalar parameters does apply here as well. The value of multi-dimensional parameters returned by system differs for HTML and the other output formats. In the case of HTML the value column will show two buttons **Download** (Sect. 5.2) and **Plot** (Sect. 5.3) whilst for the other formats the value contains the name of the dataset. Here, the hierachy of the Level-x $(1 \leq x \leq 4)$ scheme is reflected in an assumed filesystem directory structure. In addition, each dataset name possesses a unique, three-digit, zero-padded version number and the extension `.fits`. As an example, the reflectivity of the Astro telescope mirrors as a function of wavelength is encoded in a FITS dataset. The rendering of this parameter in Java reads:

```
public static final String MIRROR_REFLECTIVITY = "Astro/Mirror_Reflectivity_001.fits";
```

5.2 Downloading

In HTML the value column shows a button labelled **Download**. As the name suggests this can be used to make a copy of the datasets to the local disk. Pushing this button shall bring up a file selection dialog with a filled in default value for the name. Confirming this shall then make the copy. Note that only the basename of the dataset without any subdirectory structure will be shown by default, e.g., `Mirror_Reflectivity_001.fits` for the version 1 of the Astro telescope mirror reflectivity dataset.

5.3 Plotting

For getting a first impression of the contents, a quick-look visualisation tool is available: In HTML the value column contains a button **Plot** which loads a small applet into the browser, deploys this in a new window, downloads the corresponding dataset and displays its contents graphically. The applet is a stripped-down version of the application `SpecView`[6] developed by STScI for visualising spectral data from the Hubble Space Telescope. An extensive on-online manual can be found under `http://specview.stsci.edu/javahelp/Main.html` however please note that the applet version of `SpecView` is lacking certain features of the full application. This has been done in the interest of minimising the applet size and, hence, optimising the download time. All the essential features have been retained though, i.e., the

[5] `http://fits.gsfc.nasa.gov`
[6] `http://www.stsci.edu/resources/software_hardware/specview`

ability to plot any column against each other, zooming in/out, panning, printing, change axis units interactively, etc.

The usage of the applet itself is fairly straightforward and the function of the GUI elements should be obvious. The following table lists a few of the less-intuitive features and their activation:

action	function
single left-mouse click on displayed data curve	display data set name
double left-mouse click on displayed data curve	display FITS header contents and raw data in tabular form
singe left-mouse click in corner area (pointer changes to a hand)	pop-up dialog to change axes styles (logarithmic/linear)

Please note that currently only the visualisation of scalar table data is possible. The attempt to plot the contents of an image dataset will lead to the applet just displaying an error message.

5.4 Mirroring

The downloading of several or all multidimensional datasets through the above described method (Sect. 5.2) appears cumbersome. A much more convenient way is to obtain the data through either simple anonomous `ftp` from the URL
`ftp://ftp.rssd.esa.int/pub/Gaia/data/CalParam`
or via `wget` (see Sect. 3.2) with corresponding command line options. The following command will download the entire tree from the server and put it under the directory `/Gaia/paramdb/multidim` on the local host:

```
wget -r -m -nH --directory-prefix=Gaia/paramdb/multidim --cut-dirs=4 \
    ftp://ftp.rssd.esa.int/pub/Gaia/data/CalParam
```

`wget` has several advantages over conventional `ftp`, e.g. it can be used to download entire directory trees with a single command and in fact can be set up to really mirror the remote directory. For this the above command should be set up as a `crontab` job which executes at regular intervals, e.g. once per day:

```
# Perform a wget every day at 3am
0 3 * * * wget -r -m -nH --directory-prefix=Gaia/paramdb/multidim \
    --cut-dirs=4 ftp://ftp.rssd.esa.int/pub/Gaia/data/CalParam
```

This keeps the contents of the local directory `/Gaia/paramdb/multidim` in perfect sync with the corresponding remote one, i.e. files get deleted when the remote counterpart is removed, etc.

5.5 Usage

For using the datasets locally from code a suitable FITS reader has to be used. For all the languages supported by the database there exists at least one library that can be availed. Under `http://fits.gsfc.nasa.gov/fits_libraries.html` a quite comprehensive list of available open-source FITS libraries in the various languages can be found. For C/C++ a popular one is `cfitsio`[7] but it merely provides a very low-level interface. The following is an example of using `cfitsio` from C++ to open the Astro mirror reflectivity dataset (the root of the local copy of the dataset repository is supposed to be contained in the environment variable `GAIA_DATA_ROOT`):

```
//
//  Demonstration of opening and closing multidimensional dataset
//
#include <iostream>
#include <cstdlib>
#include <string>

#include "fitsio.h"
#include "GaiaParam.h"

using namespace std;
using namespace GaiaParam;

const char *const DATA_ROOT = "GAIA_DATA_ROOT";

int main()
{
    const char *root = getenv(DATA_ROOT);
    if (!root) {
        cerr<<DATA_ROOT<<" not set\n";
        exit(1);
    }

    string fname = (string)root + '//' +
                (string)Astro::MIRROR_REFLECTIVITY();

    fitsfile *fptr;
    int status;
    fits_open_file(&fptr, fname.c_str(), READONLY, &status);
    //
    //  do something with the file here
    //
    fits_close_file(fptr, &status);

    return status;
}
```

A Technical details

This short section is intended to provide a quick overview of the architecture and some important implementation aspects of the system. The principles of the employed basic tools

[7] `http://heasarc.gsfc.nasa.gov/docs/software/fitsio/fitsio.html`

and techniques (such as XSLT) are not explained in detail here as ample documentation on each of those can be found elsewhere.

A.1 Overview

The very core of the system is a central, relational, SQL-based database system which holds the parameters and all associated attributes in a collection of logically linked tables. The widely-used, open-source system MySQL[8] has been chosen as baseline but it is believed that any other SQL-based databases (such as Oracle) would be suitable as well. The online user is not directly exposed to the SQL-level but interacts with the system through either a dynamically generated query form (Sect. 3.1.1) or a tree-like browse interface (Sect. 3.1.2). Both of these interfaces are realized by a central software layer implemented in the PHP[9] scripting language. The software serves three main purposes:

1. Dynamically creating the user interfaces (query form and tree-view)
2. Generating SQL-queries to the database system from the contents of the query form
3. Receiving matching result sets from the database and rendering the results into the desired output format.

The result rendering is a two-stage process. First, the query results from the database are converted on the fly into an in-memory XML[10] structure by means of the so-called DOM facilities in PHP. From there, the rendering into the final output formats is driven by a collection of XSLT stylesheets on the server side. There is exactly one stylesheet per supported output format, one of which is XML itself.

A.2 Sequence of events in online query

The working principles of the application can be explained best by describing the sequence of events taking place during a typical online query. Fig. 5 shows the main elements represented by boxes and the arrows indicate data flows between them:

[8] http://www.mysql.com
[9] http://www.php.net
[10] http://www.w3.org/XML

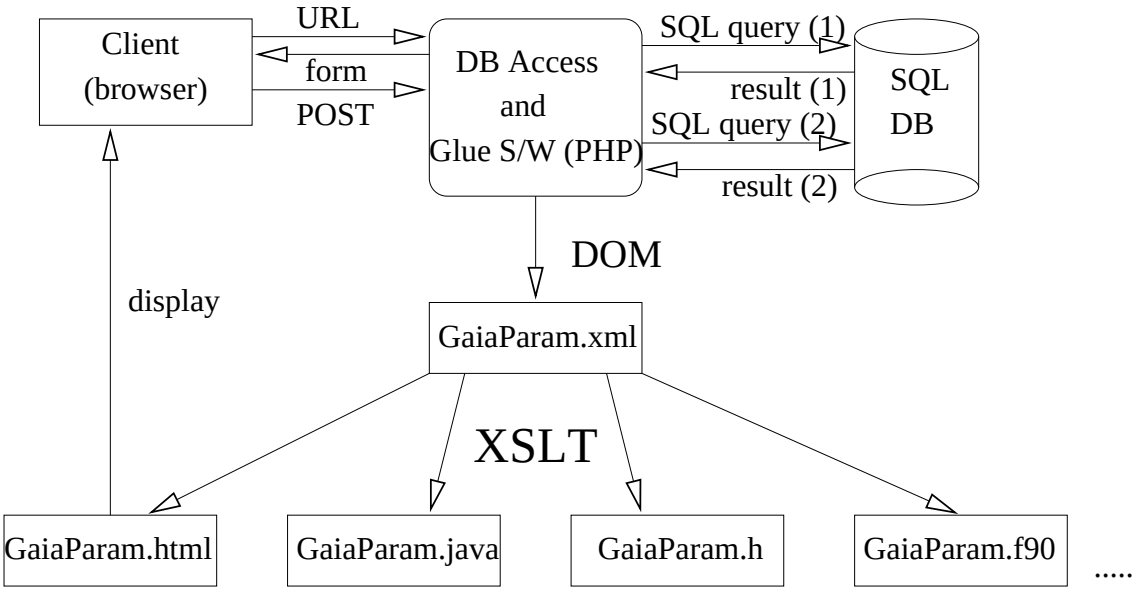

Figure 5: Relationships and interactions between the various system elements.

A remote user accessing the system through a Web browser is causing the following chain of events:

1. The client browser is accessing the query form via the corresponding URL [`URL`]
2. The query form request is causing the software to generate an SQL query [`SQL query (1)`] to the database for certain fixed data values (e.g. the Level-1 – Level-4 names, Kind names etc.)
3. The software is reading the results [`result (1)`] and constructs dynamically the query form with them
4. The complete query form is sent to the client [`form`]
5. The user fills out the form (or leaves it as is) and requests a database query by pushing the **Search** button. This sends the form data back to the server in a HTTP POST request [`POST`]
6. The software reads and interprets the form data and constructs a corresponding SQL query string and executes the query [`SQL query (2)`]
7. The result set (can be empty) is obtained from the database ... [`result (2)`]
8. ... and converted into an in-memory XML structure, the file representation of which is described in Sect. 4.2. This uses the so-called XML DOM facilities available from PHP [`DOM`]
9. Once the XML structure (in the figure illustrated as `GaiaParam.xml`) has been built the rendition to other output formats is done by means of XSLT[11] processing on the server [`XSLT`]

[11] `http://www.w3.org/Style/XSL`

10. The result is displayed in the browser window [`display`]

Note that using XML as intermediate data format is a very flexible approach since it allows conversion to almost any other format through the XSLT technique. Even client-side XSLT rendering of XML is possible (users wishing to experiment with that must enable XSLT processing in their browser). All XSLT stylesheets that control the conversion to the supported output formats are publicly accessible in the directory `access/xslt` under the URL given in Sect. 3. They may be used as templates for special client-side XSLT rendering of the XML output provided by the current system.

B Acronyms

ASCII	: American Standard Code for Information Interchange
CSV	: Comma-separated values
DOM	: Document Object Model
ESA	: European Space Agency
GSFC	: Goddard Space Flight Center
HTML	: Hyper Text Markup Language
NASA	: National Aeronautics and Space Administration
PDF	: Portable Document Format
SQL	: Structured Query Language
STScI	: Space Telescope SCience Institute
URL	: Uniform Resource Locator
XDF	: eXtensible Data Format
XML	: eXtensible Markup Language
XSLT	: eXtensible Stylesheet Language Transformations

C Acknowledgement

The `SpecView` applet is a product of the Space Telescope Science Institute, which is operated by AURA for NASA. We thank Ivo Busko (STScI) for his efforts of preparing the `SpecView` applet that is used by the Gaia Parameter Database system to visualise multi-dimensional datasets.

References

[1] Jos de Bruijne, Uwe Lammers, and Michael Perryman. A proposal for a GAIA parameter database. Technical report, ESA, June 2003. GAIA-JdB-007.