libVOTable Reference Manual

0.1

Generated by Doxygen 1.3.5

Thu May 12 22:26:52 2005

Contents

Chapter 1

libVOTable Main Page

1.1 Abstract

The libVOTable library primary task is to quickly parse big VOTABLE files that use a large amount of memory, for example, catalogues produced by big surveys. The libVOTable library has functionalities to handle VOTABLE files associated to big data bulks. This is an optimised high-speed library, especially designed for memory saving. See the `Introduction` fore more details. This TERAPIX tool is available to the community under the GPL licence as libVOTable.

1.2 Development

libVOTable is not yet finished. The following "to do list" indicates we would like to implement

Todo

Parse all elements and attributes given by VOTABLE
Check VOTABLE validity
Implementing other parsing methods

1.3 Documentation

This is the reference documentation of the libVOTable Library. This documentation has been automatically generated from the header file **votable.h**(p. 34), using the tool `doxygen`. It contains a detailed description of all functions of the libVOTable Library.

You can easily navigate through the documentation pages, using the menu above.

To get started, you may first check the list of `available modules`.

1.4 Contact

libVOTable was developed by Jean-Christophe Malapert `malapert@iap.fr`

Chapter 2

libVOTable Module Index

2.1 libVOTable Modules

Here is a list of all modules:

Chapter 3

libVOTable Data Structure Index

3.1 libVOTable Data Structures

Here are the data structures with brief descriptions:

Chapter 4

libVOTable File Index

4.1 libVOTable File List

Here is a list of all files with brief descriptions:

Chapter 5

libVOTable Page Index

5.1 libVOTable Related Pages

Here is a list of all related documentation pages:

Chapter 6

libVOTable Module Documentation

6.1 Introduction to the libVOTable library

The VOTable Library consists in a **single header file votable.h**(p. 34) providing a set of C functions that can be used in your own sources, to manage memory, parse document. It is portable , efficient and simple to use.

6.1.1 Library structure

The file **votable.h**(p. 34) contains all the functions that compose the library itself. It is organized as follows :

- Define indicating the value returned by the library.

```
#define RETURN_OK 0
#define RETURN_ERROR 1
#define EXIT_MEMORY 2
#define EXIT_READING 3
```

- All functions for memory managing
- All functions to extract attributes and values in VOTable format

To use this library in your own C code, include the following line :

```
#include "votable.h"
```

6.1.2 How to parse a VOTable file with the libVOTable

To parse a VOTable file with the libVOTable, follow the instructions :

- First, initialise memory and structures with **Init_VO_Parser**(p. 18) function.
- Then use **Extract_VO_Fields**(p. 20) to load in memory all attributes of each **FIELD** elements. Browse memory with a pointer and write the **returned** FIELD position in a array. This array contains all data positions that you want to extract in **TABLEDATA** element.

- Once it is filled, the array can be provided as a parameter to **Extract_VO_Table-Data**(p. 21) function, that it will return the data back
- If you want go on to parse (if it is possible) or jump a part of the file, you can do so the **Move_to_Next_VO_Fields**(p. 23) and **Move_to_Next_VO_Table**(p. 24) functions
- When you are at the end with the VOTable parser, don't forget to free memory ;-)

To easier the functions usage, they have been grouped together into modules in the documentation. These modules are :

- Initialization and memory management
- Extract Attributes and values from a VOTABLE element
- Jump to a VOTABLE tag

6.1.3 How to compile ?

Thanks to its packaging, the libVOTable library is a very light and user-friendly library . The easiest way to compile your own code **test.c** with the libVOTable library is to type :

```
libtool gcc -o test -static test.c libvotable.la -I/usr/local/include/libxml2 -lxml2
```

6.1.4 What's next ?

If you are ready to get more, and to start writing more serious programs with libVOTable, you are invited to go to the **Tutorial : Getting Started**(p. 13) section.

6.2 Tutorial : Getting Started

Assume you need to read the following VOTable file :

```
<?xml version="1.0"?>
<VOTABLE version="1.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="http://www.ivoa.net/xml/VOTable/VOTable/v1.1">
  <DESCRIPTION>Data to import to skywatcher</DESCRIPTION>
  <RESOURCE>
    <PARAM name="Instrument" datatype="char" arraysize="*" value="TOTO">
      <DESCRIPTION>
        This parameter is designed to store intrument's name
      </DESCRIPTION>
    </PARAM>
    <TABLE name="SpectroLog">
      <FIELD name="Target" ucd="meta.id" datatype="char" arraysize="30*"/>
      <FIELD name="Instr" ucd="instr.setup" datatype="char" arraysize="5*"/>
      <FIELD name="Dur" ucd="time.expo" datatype="int" width="5" unit="s"/>
      <FIELD name="Spectrum" ucd="meta.ref.url" datatype="float" arraysize="*"
             unit="mW/m2/nm" type="location">
        <DESCRIPTION>Spectrum absolutely calibrated</DESCRIPTION>
        <LINK type="location"
              href="http://ivoa.spectr/server?obsno="/>
      </FIELD>
      <DATA><TABLEDATA>
        <TR><TD>NGC6543</TD><TD>SWS06</TD><TD>2028</TD><TD>01301903</TD></TR>
        <TR><TD>NGC6543</TD><TD>SWS07</TD><TD>2544</TD><TD>01302004</TD></TR>
      </TABLEDATA></DATA>
    </TABLE>
  </RESOURCE>
</VOTABLE>
```

The code below that, uses the libVOTable library, does the task :

```
#include "votable.h"

int main() {
  xmlTextReaderPtr reader;
  list_field *vfield_move;
  list_tabledata *vtabledata_move;
  VOTable votable;
  int nbFields;
  int *columns;
  char file[50]="votable.xml";
  reader = Init_VO_Parser(file,&votable);

  Extract_Att_VO_Table(reader,&votable);
  printf("Table Attribute=%s\n\n",votable.table->name);

  Extract_VO_Fields(reader,&votable,&nbFields,&columns);
  for(vfield_move=votable.field;vfield_move!=NULL;vfield_move=vfield_move->next) {
    printf("name=%s\nucd=%s\ndatatype=%s\narraysize=%s\ntype=%s\nwidth=%s\nunit=%s\n\n",
           vfield_move->name,
           vfield_move->ucd,
           vfield_move->datatype,
           vfield_move->arraysize,
           vfield_move->type,
           vfield_move->width,
           vfield_move->unit);
    if(xmlStrcmp(vfield_move->ucd,"meta.id") == 0)
      columns[0] = vfield_move->position;
    if(xmlStrcmp(vfield_move->ucd,"meta.ref.url") == 0)
      columns[1] = vfield_move->position;
 }
```

```
 Extract_VO_TableData(reader,&votable, nbFields, columns);
 for(vtabledata_move=votable.tabledata;vtabledata_move!=NULL;vtabledata_move=vtabledata_move->next) {
   printf("All values=%s\n",vtabledata_move->value);
   if (vtabledata_move->colomn == columns[0])
     printf("ucd=meta.id value=%s\n",vtabledata_move->value);
   if (vtabledata_move->colomn == columns[1])
     printf("ucd=meta.ref.url value=%s\n",vtabledata_move->value);
 }

  if (Free_VO_Parser(reader,&votable,&columns) == 1)
    fprintf(stderr,"memory problem\n");
  return 0;
}
```

The program output is :

```
Table Attribute=SpectroLog

name=Spectrum
ucd=meta.ref.url
datatype=float
arraysize=*
type=location
width=(null)
unit=mW/m2/nm

name=Dur
ucd=time.expo
datatype=int
arraysize=(null)
type=(null)
width=5
unit=s

name=Instr
ucd=instr.setup
datatype=char
arraysize=5*
type=(null)
width=(null)
unit=(null)

name=Target
ucd=meta.id
datatype=char
arraysize=30*
type=(null)
width=(null)
unit=(null)

All values=01302004
ucd=meta.ref.url value=01302004
All values=NGC6543
ucd=meta.id value=NGC6543
All values=01301903
ucd=meta.ref.url value=01301903
All values=NGC6543
ucd=meta.id value=NGC6543
```

Detailled description of each task, line by line :

```
#include "votable.h"
```

Include the main and only header file of the libVOTable library.

```
int main(){
```

Definition of the main function.

```
xmlTextReaderPtr reader;
```

the xmlTextReaderPtr used

```
list_field *vfield_move;
```

Pointer on FIELD datatype

```
list_tabledata *vtabledata_move;
```

Pointer on TABLEDATA datatype

```
VOTable votable;
```

VOTABLE structure declaration

```
int nbFields;
```

Number of FIELD

```
int *columns;
```

Integer array of TD column number to parse

```
char file[50]="votable.xml";
```

Filename to parse

```
reader = Init_VO_Parser(file,&votable);
```

Memory and structure initialization

```
Extract_Att_VO_Table(reader,&votable);
```

Extract TABLE attributes

```
printf("Table Attribute=%s\n\n",votable.table->name);
```

Display result

```
Extract_VO_Fields(reader,&votable,&nbFields,&columns);
```

Extract FIELD attributes

```
for(vfield_move=votable.field;vfield_move!=NULL;vfield_move=vfield_move->next) {
  printf("name=%s\nucd=%s\ndatatype=%s\narraysize=%s\ntype=%s\nwidth=%s\nunit=%s\n\n",
          vfield_move->name,
          vfield_move->ucd,
          vfield_move->datatype,
          vfield_move->arraysize,
          vfield_move->type,
          vfield_move->width,
          vfield_move->unit);
  if(xmlStrcmp(vfield_move->ucd,"meta.id") == 0)
      columns[0] = vfield_move->position;
  if(xmlStrcmp(vfield_move->ucd,"meta.ref.url") == 0)
      columns[1] = vfield_move->position;
}
```

Browse the FIELD linking list and prepare the parsing for data for which ucd are meta.id and meta.ref.url

```
Extract_VO_TableData(reader,&votable, nbFields, columns);
```

Extract data for which `ucd` are meta.id and meta.ref.url

```
for(vtabledata_move=votable.tabledata;vtabledata_move!=NULL;vtabledata_move=vtabledata_move->next) {
  printf("All values=%s\n",vtabledata_move->value);
  if (vtabledata_move->colomn == columns[0])
    printf("ucd=meta.id value=%s\n",vtabledata_move->value);
  if (vtabledata_move->colomn == columns[1])
    printf("ucd=meta.ref.url value=%s\n",vtabledata_move->value);
}
```

Browse the TABLEDATA linking list and display parsing result

```
if (Free_VO_Parser(reader,&votable,&columns) == 1)
   fprintf(stderr,"memory problem\n");
```

Free parser memory

6.3 Initialization and memory management

Functions

- xmlTextReaderPtr **Init_VO_Parser** (const char filename, **VOTable** votablePtr)

  Parser initialization
  PURPOSE : Initializing of VOTable structure and creating xmlTextReader pointer.

- int **Free_VO_Parser** (xmlTextReaderPtr reader, **VOTable** votablePtr, int column)

  VO_Parser memory free
  PURPOSE : Free VOTable structure, column and xmlTextReader.

6.3.1 Detailed Description

Parser initializing and free memory

6.3.2 Function Documentation

6.3.2.1 int Free_VO_Parser (xmlTextReaderPtr *reader*, VOTable *votablePtr*, int *column*)

VO_Parser memory free

PURPOSE : Free VOTable structure, column and xmlTextReader.

Parameters:

reader Pointer on xmlTextReader

votablePtr Pointer on VOTable structure

column Array representing the columns position to extract embedded in TABLEDATA

Returns:

RETURN_OK : free successed
RETURN_ERROR : xmlTextReaderClose failed

```
425                                {
426   list_tabledata *vtabledata_move, *tmpPtr_tabledata;
427
428   /* Cleanup memory */
429    if (votablePtr->field != NULL)
430     Free_VO_Fields(votablePtr->field,column);
431   if (votablePtr->table != NULL)
432     Free_VO_Table(votablePtr->table);
433   if (votablePtr->tabledata != NULL)
434     Free_VO_Tabledata(votablePtr->tabledata);
435
436   xmlFreeTextReader(reader);
437
438   /*
439    * Cleanup function for the XML library.
440    */
441   xmlCleanupParser();
442   /*
443    * this is to debug memory for regression tests
444    */
```

```
445   xmlMemoryDump();
446
447   return(RETURN_OK);
448 }
```

6.3.2.2 xmlTextReaderPtr Init_VO_Parser (const char *filename*, VOTable *votablePtr*)

Parser initialization

PURPOSE : Initializing of VOTable structure and creating xmlTextReader pointer.

Parameters:

filename VOTABLE filename

votablePtr Pointer on VOTable structure

Exceptions:

EXIT_READING can't read filename and exit program

Returns:

xmlTextReaderPtr Pointer on xmlTextReader
_VOTable(p. 30) structure initialized

```
320                                                 {
321
322   xmlTextReaderPtr reader;
323
324   /* Initialisation linking lists */
325   votablePtr->field = NULL;
326   votablePtr->tabledata = NULL;
327   votablePtr->table = NULL;
328
329   /* Init xml Memory */
330   xmlInitMemory();
331
332   /* Reading file */
333   if ((reader = xmlReaderForFile(filename, NULL, 0)) == NULL) {
334     fprintf(stderr,"xmlReaderForFile failed\n");
335     exit(EXIT_READING);
336   }
337
338   return(reader);
339 }
```

6.4 Extract Attributes and values from a VOTABLE element

Functions

- void **Extract_VO_Fields** (xmlTextReaderPtr reader, **VOTable** votablePtr, int nbFields, int columns)

  Procedure to extract FIELD attribute
  PURPOSE : Free field linking list if needed, extracts FIELD attributes from VOTABLE file and stores them in VOTable structure.

- int **Extract_VO_TableData** (xmlTextReaderPtr reader, **VOTable** votablePtr, int nbcolumns, int columns)

  Procedure to extract TD tag attributes and values
  PURPOSE : Free tabledata linking list if needed, Extracts TD attributes and values from VOTABLE file and stores them in VOTable structure.

- void **Extract_Att_VO_Table** (xmlTextReaderPtr reader, **VOTable** votablePtr)

  Procedure extracting TABLE tag attributes
  PURPOSE : Free table linking list if needed, extracts TABLE attributes from VOTABLE file and stores them in VOTable structure.

6.4.1 Detailed Description

Free memory if needed, extract data from VOTABLE file and store them in the **_VOTable**(p. 30) structure

6.4.2 Function Documentation

6.4.2.1 void Extract_Att_VO_Table (xmlTextReaderPtr *reader*, VOTable *votablePtr*)

Procedure extracting TABLE tag attributes

PURPOSE : Free table linking list if needed, extracts TABLE attributes from VOTABLE file and stores them in VOTable structure.

Parameters:

reader Pointer on xmlTextReader

votablePtr Pointer on VOTable structure

Returns:

_VOTable(p. 30) structure filled
Number of records

```
169                                             {
170
171   xmlChar *name;
172   int ret;
173
174   ret = 1;
175   /* Free memory if needed */
176   if (votablePtr->table != NULL)
177     Free_VO_Table(votablePtr->table);
```

```
178   if(reader == NULL)
179     ret = xmlTextReaderRead(reader);
180   while (ret == 1) {
181     /* Reading file */
182     name = xmlTextReaderName(reader);
183      if (name == NULL)
184         name = xmlStrdup(BAD_CAST "--");
185      /* Searching TABLE tag */
186     if (xmlStrcmp(name,"TABLE") == 0
187         && xmlTextReaderNodeType(reader) == 1) {
188       votablePtr->table = insert_table(reader);
189       ret = 0;
190       xmlFree(name);
191     } else {
192       ret = xmlTextReaderRead(reader);
193       if (name!=NULL)
194         xmlFree(name);
195     }
196   }
197 }
```

6.4.2.2 void Extract_VO_Fields (xmlTextReaderPtr *reader*, VOTable *votablePtr*, int *nbFields*, int *columns*)

Procedure to extract FIELD attribute

PURPOSE : Free field linking list if needed, extracts FIELD attributes from VOTABLE file and stores them in VOTable structure.

Parameters:

reader Pointer on xmlTextReader

votablePtr Pointer on VOTable structure

nbFields FIELD tag occurence embedded in one TABLE element

columns Array representing the columns position to extract in TABLEDATA

Returns:

_**VOTable**(p. 30) structure filled
Field tag occurence in one TABLE tag
Array allocated to nbFields size and zero initialized

```
204                                         {
205
206   int ret;
207   int position;
208   int i;
209   xmlChar *name;
210
211   /* Free memory if needed */
212   if (votablePtr->field != NULL) {
213     Free_VO_Fields(votablePtr->field,columns);
214     votablePtr->field = NULL;
215   }
216
217   /* Init variable */
218   position = 0;
219   ret = 1;
220
221   /* Reading file */
222   if(reader == NULL)
223     ret = xmlTextReaderRead(reader);
```

```
224   while (ret == 1) {
225     name = xmlTextReaderName(reader);
226     if (name == NULL)
227       name = xmlStrdup(BAD_CAST "--");
228     /* Searching FIELD tag */
229     if (xmlStrcmp(name,"FIELD") == 0
230         && xmlTextReaderNodeType(reader) == 1) {
231       /* Number of FIELD met */
232       position++;
233       /* Insert in the linking list the attribute values of the element */
234       votablePtr->field = insert_field(reader,votablePtr->field,position);
235       /* go on reading */
236       ret = xmlTextReaderRead(reader);
237       xmlFree(name);
238     }
239     else if(xmlStrcmp(name,"DATA") == 0
240             && xmlTextReaderNodeType(reader) == 1) {
241       ret = 0;
242       xmlFree(name);
243     }
244     else {
245       ret = xmlTextReaderRead(reader);
246       if (name != NULL)
247         xmlFree(name);
248     }
249   }
250
251   /* Memory allocation for columns in order to avoid to user to do that*/
252   QMALLOC(*columns,int,position);
253   /* Field tag number found */
254   *nbFields = position;
255   /* Initialization of columns */
256   for(i=0;i<position;i++)
257     (*columns)[i] = 0;
258 }
```

6.4.2.3 int Extract_VO_TableData (xmlTextReaderPtr *reader*, VOTable *votablePtr*, int *nbcolumns*, int *columns*)

Procedure to extract TD tag attributes and values

PURPOSE : Free tabledata linking list if needed, Extracts TD attributes and values from VOTABLE file and stores them in VOTable structure.

Parameters:

reader Pointer on xmlTextReader

votablePtr Pointer on VOTable structure

nbcolumns Field tag occurence embedded in one TABLE element

columns Array representing the columns position to extract in TABLEDATA tag

Returns:

_**VOTable**(p. 30) structure filled
Number of records

```
265                                     {
266
267   xmlChar *name;
268   int column_number;
269   int ret,cnt,nblines;
270   int *pinit;
```

```
271
272   nblines = 0;
273   /* Free memory if needed */
274   if (votablePtr->tabledata != NULL)
275     Free_VO_Tabledata(votablePtr->tabledata);
276   /* Initialization */
277   ret = 1;
278   column_number = 0;
279   pinit = columns;
280   if(reader == NULL)
281     ret = xmlTextReaderRead(reader);
282   while (ret == 1) {
283     name = xmlTextReaderName(reader);
284     if (name == NULL)
285       name = xmlStrdup(BAD_CAST "--");
286     /* Search TD node*/
287     if (xmlStrcmp(name,"TD") == 0
288         && xmlTextReaderNodeType(reader) == 1) {
289       /* Retrieve TD tag value */
290       ret = xmlTextReaderRead(reader);
291       xmlFree(name);
292       column_number++;
293       /* retrieve all data for columns selected */
294       for(cnt=0;cnt<nbcolumns;cnt++)
295         if (columns[cnt] == column_number)
296           votablePtr->tabledata = insert_tabledata(reader,votablePtr->tabledata,column_number);
297
298       columns = pinit;
299       /* Start a TR tag */
300       if (column_number == nbcolumns) {
301         column_number = 0;
302         nblines++;
303       }
304     } else if(xmlStrcmp(name,"TABLEDATA") == 0
305               && xmlTextReaderNodeType(reader) == 15) {
306         ret = 0;
307         xmlFree(name);
308     }
309     else {
310       ret = xmlTextReaderRead(reader);
311       if (name != NULL)
312         xmlFree(name);
313     }
314   }
315   return(nblines);
316 }
```

6.5 Jump to a VOTABLE tag

Functions

- int **Move_to_Next_VO_Fields** (xmlTextReaderPtr reader)

  Move reader to the first FIELD tag contained in the next TABLE tag
  PURPOSE : Try to move to the first FIELD tag embedded in the next TABLE tag.

- int **Move_to_Next_VO_Table** (xmlTextReaderPtr reader)

  Try to move to the next TABLE tag
  PURPOSE : Try to move to the next TABLE tag.

6.5.1 Detailed Description

Functions to jump to a VOTABLE tag

6.5.2 Function Documentation

6.5.2.1 int Move_to_Next_VO_Fields (xmlTextReaderPtr *reader*)

Move reader to the first FIELD tag contained in the next TABLE tag

PURPOSE : Try to move to the first FIELD tag embedded in the next TABLE tag.

Parameters:
reader Pointer on xmlTextReader

Exceptions:
RETURN_ERROR can't move to the next TABLE tag
Move reader to the end of file

Returns:
RETURN_OK Move reader to the first FIELD tag of the next TABLE tag

```
113                                                 {
114
115   int ret;
116   xmlChar *name;
117
118   ret = 1;
119
120   /* Reading file */
121   ret = xmlTextReaderRead(reader);
122   while (ret == 1) {
123     name = xmlTextReaderName(reader);
124     /* Searching FIELD tag */
125     if (xmlStrcmp(name,"FIELD") == 0
126         && xmlTextReaderNodeType(reader) == 1) {
127       xmlFree(name);
128       return(RETURN_OK);
129     } else {
130       ret = xmlTextReaderRead(reader);
131       if (name!=NULL)
132         xmlFree(name);
133     }
```

```
134
135   }
136   return(RETURN_ERROR);
137 }
```

6.5.2.2 int Move_to_Next_VO_Table (xmlTextReaderPtr *reader*)

Try to move to the next TABLE tag

PURPOSE : Try to move to the next TABLE tag.

Parameters:

reader Pointer on xmlTextReader

Exceptions:

RETURN_ERROR can't move to the first FIELD embedded in the next TABLE element
Move reader to the end of file

Returns:

RETURN_OK Move reader to the next TABLE tag

```
141                                                 {
142
143   int ret;
144   xmlChar *name;
145
146   ret = 1;
147   /* Reading file */
148   ret = xmlTextReaderRead(reader);
149   while (ret == 1) {
150     name = xmlTextReaderName(reader);
151     /* Searching TABLE tag */
152     if (xmlStrcmp(name,"TABLE") == 0
153         && xmlTextReaderNodeType(reader) == 1) {
154       xmlFree(name);
155       return(RETURN_OK);
156     }
157     else {
158       ret = xmlTextReaderRead(reader);
159       if (name!=NULL)
160         xmlFree(name);
161     }
162   }
163   return(RETURN_ERROR);
164 }
```

Chapter 7

libVOTable Data Structure Documentation

7.1 _list_field Struct Reference

Structure dedicated to FIELD tag.

`#include <votable.h>`

Data Fields

- xmlChar **ID**

  ID used : implied.

- xmlChar **unit**

  unit used : implied

- xmlChar **datatype**

  datatype used : implied
  among : boolean, bit, unsignedByte, short ,int, long, char unicodeChar, float, double, floatComplex or doubleComplex

- xmlChar **precision**

  value precision used : implied

- xmlChar **width**

  value width used : implied

- xmlChar **ref**

  ref attribute used : implied

- xmlChar **name**

  name attribute used : implied

- xmlChar **ucd**

unified content type attribute used : implied

- xmlChar **arraysize**

  arraysize used : implied

- xmlChar **type**

  type used among : implied
  among : hidden, no_ query or trigger

- int **position**

  position
  position of the FIELD element in respect to others ones

- **list_field next**

  address of the next element of the list

7.1.1 Detailed Description

Structure dedicated to FIELD tag.

Structure containing all attributes allowed for FIELD tag.

In order to save all this structure type, we create a linking list of this one.

7.1.2 Field Documentation

7.1.2.1 xmlChar _list_field::arraysize

arraysize used : implied

7.1.2.2 xmlChar _list_field::datatype

datatype used : implied

among : boolean, bit, unsignedByte, short ,int, long, char unicodeChar, float, double, floatComplex or doubleComplex

7.1.2.3 xmlChar _list_field::ID

ID used : implied.

7.1.2.4 xmlChar _list_field::name

name attribute used : implied

7.1.2.5 list_field _list_field::next

address of the next element of the list

7.1.2.6 int _list_field::position

position

position of the FIELD element in respect to others ones

7.1.2.7 xmlChar _list_field::precision

value precision used : implied

7.1.2.8 xmlChar _list_field::ref

ref attribute used : implied

7.1.2.9 xmlChar _list_field::type

type used among : implied

among : hidden, no_query or trigger

7.1.2.10 xmlChar _list_field::ucd

unified content type attribute used : implied

7.1.2.11 xmlChar _list_field::unit

unit used : implied

7.1.2.12 xmlChar _list_field::width

value width used : implied

The documentation for this struct was generated from the following file:

- **votable.h**

7.2 _list_table Struct Reference

Structure dedicated to TABLE tag.

`#include <votable.h>`

Data Fields

- xmlChar **ID**
  ID attribute used : implied.
- xmlChar **name**
  name attribute used : implied
- xmlChar **ref**
  ref attribute used : implied

7.2.1 Detailed Description

Structure dedicated to TABLE tag.

Structure containing all attributes allowed for TABLE tag.

In order to save all this structure type, we create a linking list of this one.

7.2.2 Field Documentation

7.2.2.1 xmlChar _list_table::ID

ID attribute used : implied.

7.2.2.2 xmlChar _list_table::name

name attribute used : implied

7.2.2.3 xmlChar _list_table::ref

ref attribute used : implied

The documentation for this struct was generated from the following file:

- **votable.h**

7.3 _list_tabledata Struct Reference

Structure dedicated to TABLEDATA tag.

`#include <votable.h>`

Data Fields

- xmlChar **value**
  TD tag value used.
- xmlChar **ref**
  TABLEDATA attribute used.
- int **colomn**
  column to parse
- **list_tabledata** **next**
  address of the next element of the list

7.3.1 Detailed Description

Structure dedicated to TABLEDATA tag.

The TABLEDATA does not contain attributes. Moreover, only the tag embedded in the TABLEDATA element contains a PCDATA value and attribute. For this reason and to reduce the number of structures, the TD structure is the TABLEDATA structure containing all attributes allowed for TD tag and its associated value. In order to save all this structure type, we create a linking list.

7.3.2 Field Documentation

7.3.2.1 int _list_tabledata::colomn

column to parse

7.3.2.2 list_tabledata _list_tabledata::next

address of the next element of the list

7.3.2.3 xmlChar _list_tabledata::ref

TABLEDATA attribute used.

7.3.2.4 xmlChar _list_tabledata::value

TD tag value used.

The documentation for this struct was generated from the following file:

- **votable.h**

7.4 _VOTable Struct Reference

Main structure.

`#include <votable.h>`

Data Fields

- **list_table table**
  structure dedicated to TABLE tag
- **list_tabledata tabledata**
  structure dedicated to TABLEDATA tag
- **list_field field**
  structure dedicated to FIELD tag

7.4.1 Detailed Description

Main structure.

Structure containing all structures relative to VOTABLE

7.4.2 Field Documentation

7.4.2.1 list_field _VOTable::field

structure dedicated to FIELD tag

7.4.2.2 list_table _VOTable::table

structure dedicated to TABLE tag

7.4.2.3 list_tabledata _VOTable::tabledata

structure dedicated to TABLEDATA tag

The documentation for this struct was generated from the following file:

- **votable.h**

Chapter 8

libVOTable File Documentation

8.1 example.c File Reference

A simple using of libVOTable.

`#include "votable.h"`

Functions

- int **main** ()

8.1.1 Detailed Description

A simple using of libVOTable.

8.1.2 Function Documentation

8.1.2.1 int main ()

```
7           {
8   xmlTextReaderPtr reader;
9   list_field *vfield_move;
10   list_tabledata *vtabledata_move;
11   VOTable votable;
12   int nbFields, process_column;
13   int *columns;
14   char file[50]="votable.xml";
15   reader = Init_VO_Parser(file,&votable);
16
17   Extract_Att_VO_Table(reader,&votable);
18   printf("Table Attribute=%s\n\n",votable.table->name);
19
20   Extract_VO_Fields(reader,&votable,&nbFields,&columns);
21   for(vfield_move=votable.field;vfield_move!=NULL;vfield_move=vfield_move->next) {
22     printf("name=%s\nucd=%s\ndatatype=%s\narraysize=%s\ntype=%s\nwidth=%s\nunit=%s\n\n",
23            vfield_move->name,
24            vfield_move->ucd,
25            vfield_move->datatype,
26            vfield_move->arraysize,
27            vfield_move->type,
```

```
28            vfield_move->width,
29            vfield_move->unit);
30     if(xmlStrcmp(vfield_move->ucd,"meta.id") == 0)
31       columns[0] = vfield_move->position;
32     if(xmlStrcmp(vfield_move->ucd,"meta.ref.url") == 0)
33       columns[1] = vfield_move->position;
34  }
35
36
37  Extract_VO_TableData(reader,&votable, nbFields, columns);
38  for(vtabledata_move=votable.tabledata;vtabledata_move!=NULL;vtabledata_move=vtabledata_move->next) {
39    printf("All values=%s\n",vtabledata_move->value);
40    if (vtabledata_move->colomn == columns[0])
41      printf("ucd=meta.id value=%s\n",vtabledata_move->value);
42    if (vtabledata_move->colomn == columns[1])
43      printf("ucd=meta.ref.url value=%s\n",vtabledata_move->value);
44  }
45
46   if (Free_VO_Parser(reader,&votable,&columns) == 1)
47     fprintf(stderr,"memory problem\n");
48   return 0;
49 }
```

8.2 votable.c File Reference

```
#include "votable.h"
```

Functions

- int **Move_to_Next_VO_Fields** (xmlTextReaderPtr reader)

  Move reader to the first FIELD tag contained in the next TABLE tag
  PURPOSE : Try to move to the first FIELD tag embedded in the next TABLE tag.

- int **Move_to_Next_VO_Table** (xmlTextReaderPtr reader)

  Try to move to the next TABLE tag
  PURPOSE : Try to move to the next TABLE tag.

- void **Extract_Att_VO_Table** (xmlTextReaderPtr reader, **VOTable** votablePtr)

  Procedure extracting TABLE tag attributes
  PURPOSE : Free table linking list if needed, extracts TABLE attributes from VOTABLE file and stores them in VOTable structure.

- void **Extract_VO_Fields** (xmlTextReaderPtr reader, **VOTable** votablePtr, int nbFields, int columns)

  Procedure to extract FIELD attribute
  PURPOSE : Free field linking list if needed, extracts FIELD attributes from VOTABLE file and stores them in VOTable structure.

- int **Extract_VO_TableData** (xmlTextReaderPtr reader, **VOTable** votablePtr, int nbcolumns, int columns)

  Procedure to extract TD tag attributes and values
  PURPOSE : Free tabledata linking list if needed, Extracts TD attributes and values from VOTABLE file and stores them in VOTable structure.

- xmlTextReaderPtr **Init_VO_Parser** (const char filename, **VOTable** votablePtr)

  Parser initialization
  PURPOSE : Initializing of VOTable structure and creating xmlTextReader pointer.

- int **Free_VO_Parser** (xmlTextReaderPtr reader, **VOTable** votablePtr, int column)

  VO_Parser memory free
  PURPOSE : Free VOTable structure, column and xmlTextReader.

8.3 votable.h File Reference

libVOTable header file

```
#include <stdio.h>
#include <stdlib.h>
#include <libxml/xmlmemory.h>
#include <libxml/xmlreader.h>
```

Data Structures

- struct **_list_field**
  Structure dedicated to FIELD tag.
- struct **_list_table**
  Structure dedicated to TABLE tag.
- struct **_list_tabledata**
  Structure dedicated to TABLEDATA tag.
- struct **_VOTable**
  Main structure.

Defines

- #define **DEBUG**(x) printf(x)
- #define **RETURN_OK** 0
  One of Codes returned by functions.
- #define **RETURN_ERROR** 1
  One of Codes returned by functions.
- #define **EXIT_MEMORY** 2
  One of Codes returned by functions.
- #define **EXIT_READING** 3
  Can't open file.
- #define **QMALLOC**(ptr, typ, nel)
  Macro for memory allocation.

Typedefs

- typedef **_list_field list_field**
- typedef **_list_tabledata list_tabledata**
- typedef **_list_table list_table**
- typedef **_VOTable VOTable**

Functions

- xmlTextReaderPtr **Init_VO_Parser** (const char filename, **VOTable** votablePtr)

  Parser initialization
  PURPOSE : Initializing of VOTable structure and creating xmlTextReader pointer.

- int **Free_VO_Parser** (xmlTextReaderPtr reader, **VOTable** votablePtr, int column)

  VO_Parser memory free
  PURPOSE : Free VOTable structure, column and xmlTextReader.

- void **Extract_VO_Fields** (xmlTextReaderPtr reader, **VOTable** votablePtr, int nbFields, int columns)

  Procedure to extract FIELD attribute
  PURPOSE : Free field linking list if needed, extracts FIELD attributes from VOTABLE file and stores them in VOTable structure.

- int **Extract_VO_TableData** (xmlTextReaderPtr reader, **VOTable** votablePtr, int nbcolumns, int columns)

  Procedure to extract TD tag attributes and values
  PURPOSE : Free tabledata linking list if needed, Extracts TD attributes and values from VOTABLE file and stores them in VOTable structure.

- void **Extract_Att_VO_Table** (xmlTextReaderPtr reader, **VOTable** votablePtr)

  Procedure extracting TABLE tag attributes
  PURPOSE : Free table linking list if needed, extracts TABLE attributes from VOTABLE file and stores them in VOTable structure.

- int **Move_to_Next_VO_Fields** (xmlTextReaderPtr reader)

  Move reader to the first FIELD tag contained in the next TABLE tag
  PURPOSE : Try to move to the first FIELD tag embedded in the next TABLE tag.

- int **Move_to_Next_VO_Table** (xmlTextReaderPtr reader)

  Try to move to the next TABLE tag
  PURPOSE : Try to move to the next TABLE tag.

8.3.1 Detailed Description

libVOTable header file

8.3.2 Define Documentation

8.3.2.1 #define DEBUG(x) printf(x)

8.3.2.2 #define EXIT_MEMORY 2

One of Codes returned by functions.

is return when there is not enough memory

8.3.2.3 #define EXIT_READING 3

Can't open file.

exit with EXIT_MEMORY value when xmlReaderForFile function failed

8.3.2.4 #define QMALLOC(ptr, typ, nel)

Value:

```
{if (!(ptr = (typ *)malloc((size_t)(nel)*sizeof(typ)))) \
                  error(EXIT_MEMORY, "Not enough memory for ", \
                        #ptr " (" #nel " elements) !");;}
```

Macro for memory allocation.

8.3.2.5 #define RETURN_ERROR 1

One of Codes returned by functions.

is returned when the function failed

8.3.2.6 #define RETURN_OK 0

One of Codes returned by functions.

is returned when the function successed

8.3.3 Typedef Documentation

8.3.3.1 typedef struct _list_field list_field

8.3.3.2 typedef struct _list_table list_table

8.3.3.3 typedef struct _list_tabledata list_tabledata

8.3.3.4 typedef struct _VOTable VOTable

Chapter 9

libVOTable Page Documentation

9.1 Todo List

page Main Page(p. 1) Parse all elements and attributes given by VOTABLE
Check VOTABLE validity
Implementing other parsing methods

Index