

# VisIVOServer 2.1

## User Guide

### Introduction

**Authors:** Becciani U., Caniglia G., Costa A., Gheller C., Krokos M., Massimino P., Sciacca E., Vitello F.

**VisIVO Server** is a suite of software tools for creating customized views of 3D renderings from astrophysical data tables. These tools are founded on the **VisIVO Desktop** functionality ([visivo.oact.inaf.it](http://visivo.oact.inaf.it)) and support the most popular Linux based platforms (e.g. [www.ubuntu.com](http://www.ubuntu.com)). Their defining characteristic is that no fixed limits are prescribed regarding the dimensionality of data tables input for processing, thus supporting very large scale datasets.

VisIVO Server websites are currently hosted by the University of Portsmouth, UK ([visivo.port.ac.uk](http://visivo.port.ac.uk)), the INAF Astrophysical Observatory of Catania, Italy ([visivo.oact.inaf.it](http://visivo.oact.inaf.it)) and in the near future by CINECA, Italy ([visivo.cineca.it](http://visivo.cineca.it)). These web sites offer data management functionality for registered users; datasets can be uploaded for temporary storage and processing for a period of up to two months. The sites can also be utilized through anonymous access in which case datasets can be uploaded and stored for a maximum of four days; to maximize available resources a limited dimensionality is only supported.

Assuming that datasets are uploaded, users are typically presented with tree-like structures (for easy data navigation) containing pointers to **files**, **tables**, **volumes** as well as **visuals**.

**Files** point to single, or possibly several (for distributed datasets), astrophysical data tables;

**Tables** are highly-efficient internal VisIVO Server data representations; they are typically produced from importing datasets uploaded by users using VisIVO Importer (see below);

**Volumes** are internal VisIVO Server data representations; they are produced either from direct importing of user datasets or by performing operations on already existing tables;

**Visuals** are collections of highly-customized, user-produced views of 3D renderings of volumes.

VisIVO Server consists of three core components: **VisIVO Importer**, **VisIVO Filter** and **VisIVO Viewer** respectively. Their functionality and usage is described in the following sections.

To create customized views of 3D renderings from astrophysical data tables, a two-stage process is employed. First, VisIVO Importer is utilized to convert user datasets into **VisIVO**

**Binary Tables** (VBTs). Then, VisIVO Viewer is invoked to display customized views of 3D renderings. As an example, consider displaying views from only three columns of an astrophysical data table supplied in ascii form, say col\_1, col\_2 and col\_3, by using the commands

```
VisIVOImporter --fformat ascii UserDataSet.txt
```

```
VisIVOViewer -x col_1 -y col_2 -z col_3 --scale --glyphs pixel VBT.bin
```

VisIVO Server is distributed with GPL V.2 License for NON COMMERCIAL use. VisIVO Server is hosted by sourceforge <https://sourceforge.net/projects/visivoserver/> and its source code is downloadable via svn:

```
svn co https://visivoserver.svn.sourceforge.net/svnroot/visivoserver/branches/1.2 visivoserver
```

Disclaimer: user data integrity is never warranted.

## VisIVO BINARY TABLE

A VisIVO Binary Table (VBT) is a highly-efficient data representation used by VisIVO Server internally. A VBT is realized through a header file (extension **.bin.head**) containing all necessary metadata, and a raw data file (extension **.bin**) storing actual data values. For example, the header may contain information regarding the overall number of fields and number of points for each field (for point datasets) or the number of cells and relevant mesh sizes (for volume datasets). The raw data file is typically a sequence of values, e.g. all X followed by all Y values.

### Header

The header file contains the following fields:

```
float | double
n1
n2 [ GeoX GeoY GeoZ DX DY DZ ]
little | big
X
Y
Z
Vx
Vy
Vz
```

- **float | double** is the data type of the storage variables used;
- **n1** denotes the number of columns (fields) in the VBT;
- **n2** denotes the number of rows in the VBT;

- **GeoX GeoY GeoZ DX DY DZ** are employed only if the VBT represents volumetric datasets. In that case GeoX, GeoY and GeoZ represent the mesh geometry, while DX, DY and DZ represent the x, y and z size of volumetric cells.
- **little | big** denotes the endianness employed in the VBT. After this field there exist  $n1$  rows that indicate the VBT columns as positions (X, Y, Z) and velocities (Vx, Vy, Vz).

## Raw

The binary file is simply a sequence of  $n1*n2$  values. In the example shown in section 2.1 all X values, then all Y values and so on.

Note:

- **$n1$**  represents the number of columns (fields) in the VBT (e.g. 6);
- **$n2$**  represents the number of elements of each field in the VBT (e.g. 262144);
- **GeoX GeoY GeoZ** represent the number of the volumetric cells in each dimension of the mesh size (used only for Volumes) (e.g. 64 64 64);
- **DX DY DZ** represent the size of each cell (used only for Volumes) (e.g. 1.0 1.0 1.0)

# VisIVO Server

## VisIVO Importer

VisIVO Importer converts user-supplied datasets into a VBT. VBTs are used by VisIVO Filters for data processing and the VisIVO Viewer for display. The conversion is independent of data dimensionality. The VisIVO Importer command flags are explained in detail in the rest of this section.

### General VisIVO Importer Syntax

```
VisIVOImporter --flag1 --flag2 ... --flagN FileName
```

or

```
VisIVOImporter parameterFile
```

### VisIVO Importer on gLite systems

VisIVO may be compiled on a gLite system. In this case the option *GLITE* must be given for compile the tool. The main purpose is to read data from catalogue and to create a VBT in the local filesystem or in the catalogue.

When running on gLite system:

- 1) the option `--VO` is to specifying the virtual organization (VO) name.
- 2) the input user file, to be imported, can be a logical filename (lfn) and must start with *lfn://*.
- 3) the output VBT, can be local (`--out` option) or the output lfn can be given. In any case the `--out` is the local filename where a temporary VBT will be created
- 4) `--lfnout`: is the output logical filename. In this case the local temporary VBT will be saved in the grid catalogue and deleted from local filesystem.
- 5) the option `--se` is to specifying the Storage Element (SE)

## VisIVO Importer General Options

**--fformat** specifies the format of input datasets;

**--out** [name] (optional) OutputFileName. The path specified with this flag is the VisIVO Importer output directory. If no path is specified the VisIVO Importer output directory is the current directory, and the default name for OutputFileName is VisIVOServerBinary.bin;

**--volume** (optional) is used to create volumes;

**--compx** [value] **--compy** [value] **--compz** [value] (optional) is used for entering geometry for volumetric datasets; if data size fits in a cubic mesh, values are computed automatically;

**--sizex** [value] **--sizey** [value] **--sizez** [value] (optional) is used for specifying volumetric cell dimensions; the default values are 1.0, 1.0 and 1.0 respectively;

**--userpwd** [username:password] (optional) is used to prescribe a username and password for accessing remote files. Ambiguous characters for the shell (i.e. \$ >, < etc..) must be given with escape character "\". For example guest\$09 must be given as *guest\\$09*.

NOTE: escape characters MUST NOT BE GIVEN using the parameterFile.

**--binaryheader** [headerfilename] (optional) is used to specify the file name of the header of VBTs; this flag is ignored in case of other formats;

**--missingvalue** [value] (optional) is used to set, to a fixed value, the missing data. If not present a default value is:-1.0918273645e+23

**--textvalue** [value] (optional) is used to set to, a fixed value, the textual data. If not present a default value is given:-1.4536271809e+15

**--bigendian** (optional) is used only for big endian Gadget and FLY input files, otherwise by default this flag is set to little endian;

**--double** (optional) is used only for the double data type of FLY input files, otherwise by default this flag is set to float;

**--npoints** [value] (optional) is used only for FLY input files to specify the number of data points.

**--history** (optional) create an XML file which contains the history of operations performed (default create hist.xml file)

**--historyfile** [filename] (optional) Change default history file name and or directory

**--VO** [value] (optional) is used to set the Virtual Organization (VO) when running on gLite grid. It is mandatory when running on gLite using catalogue.

**--se** [value] (optional) is used to set the Storage Element (SE) when running on gLite grid. Default value is DPM\_HOST ( site default storage element).

**--lfnout** [value] (optional) is used to set the logical filename (lfn) when running on gLite grid. It must be given to store produced VBT in the catalogue

The *FileName* is the local (or remote) file containing the data to be converted into a VBT. If *FileName* starts with *http://* , *ftp://* or *sftp://* the remote file is downloaded automatically. However if the *--userpwd* option is specified, the prescribed username and password are employed for remote access. Filename starting with *lfn://* are considered logical filename if VisIVO is compiled with GLITE option.

NOTE: sftp is not allowed if VisIVO is compiled with the LIGHT option: the server must have libcurl with ssl support to enable the sftp functionality and VisIVO must be compiled withno LIGHT option.

Example:

**VisIVOImporter --fformat ascii --out mytable sftp://machinename.domain/home/user/asciifile.txt.**

The sftp syntax requires the remote directory where data are located.

All downloaded files are copied temporarily into the directory given in *--out* option and are deleted automatically at the end of the import process. Under the current file directory, the file *DownloadedFilename\_VisIVO\_List* contains information on all download operations. If the *fformat* option is *binary* VisIVOImporter will attempt to download two remote files, a binary table (given as remote filename) and its associated header file (same name + ".head" extension).

# VisIVO Importer

## ParameterFile

All the following listed importers contain the --fformat code and options. The code and all the options can be given in a parameterFile

Lines starting with # are comments.

Examples of this file are the following:

```
fformat=ascii          ← a valid fformat code
out=outFilename.bin
file=asciinputFile
```

```
fformat=votablefast
out=/home/user/dataNewTable.bin
missingvalue=0.0
file=myVOtable.xml
```

```
lfnout=lfn://grid/cometa/ube/VSIimage
VO=cometa
se=inaf-se-01.ct.pi2s2.it
```

```
fformat=fly
out=FlyData.bin
double=true
npoints=1000
bigendian=true
userpwd=myusername:mypassword
file=http://remotehosts.domain.eu/directory/InputDataFile
```

### NOTE:

- 1) The parameterFile must contain the options with the same names reported in the following as "--" options.
- 2) Options with one or more parameters must be all specified after the "=" sign in the same line. (ex. field=X Y Z).
- 3) Options that do not require parameters must be given with "true" keyword (ex. double=true, bigendian=true).
- 4) The input filename has the keyword *file* (ex. file=myInputFilename).

# VisIVO Importer

## ASCII and CSV FORMAT

ASCII files are expected to be in tabular form. An ASCII file may contain values for N variables organized in columns. The columns are typically separated by whitespace characters, e.g. spaces or tabs. The first row of an ASCII file lists the N variables names explicitly. As an example, the command below produces *NewTable.bin*, *NewTable.bin.head* from *ASCIIUserFileName.txt*.

### Syntax Example

```
VisIVOImporter --fformat ascii --out /home/user/data/NewTable ASCIIUserFileName.txt
```

CSV is a delimited data format that has fields/columns separated by the comma character and records/rows separated by newlines. Fields that contain a special character (such as comma, newline, or double quote) must be enclosed in double quotes. However, if a line contains a single entry that happens to be the empty string, it may be enclosed in double quotes. If a field's value is a double quote character, this is dealt with by placing another double quote character next to it. The CSV file format does not require a specific character encoding, byte order, or line terminator format. As an example, the command below produces the files *NewTable.bin* and *NewTable.bin.head* from the user-supplied CSV file *CSVUserFileName.txt*.

### Syntax Example

```
VisIVOImporter --fformat csv --out /home/user/data/NewTable CSVUserFileName.txt
```

### NOTE

The Importer automatically skips all the lines starting with # character. If the first line contains column names starting with #, this character will be removed, and the columns names will be given without it.



# VisIVO Importer

## BINARY FORMAT

The binary format is supposed to be the Internal Binary Table

### Syntax Example:

**VisIVOImporter --fformat binary --out /home/user/data/NewTable BinaryUserFilename.bin**

Assuming that a VBT is to be processed, the previous command produces *NewTable.bin* and *NewTable.bin.head* from *VBUserFileName.bin*. Note that the files *VBUserFileName.bin* and *VBUserFileName.bin.head* must be (either local or remote) existing files. Also, if the `--binaryheader` option is prescribed, a header file with the specified filename must exist. In case this header file resides remotely, a complete copy is created within the VisIVO Server output directory.

### Note

This command is useful for changing the endianism of a binary table. An input big endian table is transformed in a little endian table if system where VisIVOImporter is running is a little endian system and viceversa.

# VisIVO Importer

## FLY FORMAT

FLY is code that uses the tree N-body method, for three-dimensional self-gravitating collisionless systems evolution. FLY is a fully parallel code based on the tree Barnes-Hut algorithm; periodical boundary conditions are implemented by means of the Ewald summation technique.

FLY is based on the one-side communication paradigm for sharing data among processors, accessing remotely private data without synchronism.

The FLY output format is a binary sequence of values of  $n$  data points as follows:  $X_1, Y_1, Z_1, X_2, Y_2, Z_2, \dots, X_n, Y_n, Z_n, V_{x1}, V_{y1}, V_{z1}, V_{x2}, V_{y2}, V_{z2}, \dots, V_{xn}, V_{yn}, V_{zn}$ . As an example, the command below produces *NewTable.bin* and *NewTable.bin.head* from FLYUserFile which is a FLY file using double data types, containing a total of 2000000 data points.

### Syntax Example

```
VisIVOImporter --fformat fly --out /home/user/data/NewTable --double --npoints
2000000 FLYUserFile
```

The FLY format also allows the download of elements given in a descriptor file (.desc extension):

```
flyDesc      (type of the Fly descriptor)
2m_test      (ID)
double       (data type)
time         (ID for snapshot sequence)
2097152      (number of points on each snapshot)
50 50 50     (box dimension in the proper unit)
1            ("l" or "b" for data endianism)
0.0 out_scdm_0.0000 (time tag and snapshot filename sequence)
1.0 out_scdm_1.0000
2.0 out_scdm_2.0000
```

In this case the flags `--npoints` and `--double` must not be given as values are read automatically from the descriptor file. Each listed file (`out_scdm_#` in this case) produces a VBT.

### Syntax Example

```
VisIVOImporter --fformat fly --out /home/user/data/NewTable FLYUserFile.desc
```

Note that names of output files will be determined by using the `--out` parameter concatenated first by a `"_"` and then by the listed filename. As an example, if `--out /tmp/pippo` is prescribed, output filenames will be `/tmp/pippo_out_scdm_0.000.bin` (and one with the extension `.bin.head`). On the other hand, if `--out` option is not prescribed,

output filenames will be `./VisIVOServerBinaryout_scdm_0.000.bin` (and one with extension `.bin.head`).

# VisIVO Importer

## FITS Table FORMAT

The definition of FITS is a codification into a formal standard, by the NASA/Science Office of Standards and Technology (NOST), of the FITS rules (<http://fits.gsfc.nasa.gov>) endorsed by the IAU. FITS supports tabular data with named columns and multidimensional rows. Both binary and ASCII FITS table versions have been specified. The data in a column of a FITS table can be in a different format from the data in other columns. Together with the ability to string multiple header/data blocks together, by using FITS files it is possible to represent entire relational databases. As an example, the command below produces *NewTable.bin*, *NewTable.bin.head* from FITSTableUserFile.

### Syntax Example

```
VisIVOImporter --fformat fitstable --out /home/user/data/NewTable
FITSTableUserFile
```

# VisIVO Importer

## FITS Image FORMAT

The Definition of FITS is a codification into a formal standard, by the NASA/Science Office of Standards and Technology (NOST), of the FITS rules endorsed by the IAU. FITS image headers can contain information about one or more scientific coordinate systems that are overlain on the image itself. Images contain an implicit Cartesian coordinate system that describes the location of each pixel in the image, but scientific uses generally require working in 'world' coordinates, for example the celestial coordinate system.

### Under Development

# VisIVO Importer

## GADGET FORMAT

GADGET is freely-available code for cosmological N-body/SPH simulations on massively parallel computers with distributed memory. GADGET uses an explicit communication model that is implemented with the standardized MPI communication interface. The code can be run on essentially all supercomputer systems presently in use, including clusters of workstations or individual PCs.

VisIVO Importer will produce a VBT for each species in the gadget file supplied. As an example, the command below will produce the files *NewTableHALO.bin*, *NewTableHALO.bin.head*, and *NewTableGAS.bin*, *NewTableGAS.bin.head* from the GADGET file *GadgetUserFile* if we assume that it contains only two species, namely halo and gas particles.

### Syntax Example

```
VisIVOImporter --fformat gadget --out /home/user/data/NewTable GadgetUserFile
```

# VisIVO Importer

## HDF5 FORMAT

The hierarchical Data Format is a library and multi-object file format for the transfer of graphical and numerical data between computers. It was created by the NCSA, but is currently maintained by The HDF Group. The freely available HDF distribution consists of the library, command-line utilities, test suite source, Java interface, and the Java-based HDF Viewer (HDFView). HDF supports several different data models, including multidimensional arrays, raster images, and tables. Each defines a specific aggregate data type and provides an API for reading, writing, and organizing data and metadata. New data models can be added by the HDF developers or users.

### Syntax Example

```
VisIVOImporter --fformat hdf5 --datasetlist dataset1 dataset2 --hyperslab dataset2  
10,10 100,100 --out /home/user/data/NewTable HDF5UserFile
```

```
VisIVOImporter --fformat hdf5 --datasetlist vol1 vol2 --hyperslab vol1 0,0,0
10,10,10 vol2 5,5,5 10,10,10 --compx 10 --compy 10 --compz 10 --out
/home/user/data/NewTable HDF5UserFile
```

This importer can be used with the datasetlist option. When the datasetlist option is not given all datasets in the hdf5 will be considered forming the VBT.

## Dataset

**--datasetlist nameOfDataset1 nameOfDataset2 nameOfDataset3 ...**

If the parameterfile is used the file can contain more rows listing datasets. The parameter file must have only one dataset in the datasetlist parameter. More rows with datasetlist parameter must be given to import more than one dataset.

## Hyperslab

The hyperslab option must be given as follows:

**--hyperslab nameOfDataset offset count**

where offset and counts are the same of the hdf5 file and must be comma separated values. If count exceeds the size of dataset, it is automatically adjusted up to the end of hyperslab. If offset and/or count contains lower values than the dataset dimension, not specified **offset** are put to **0** and **count** to the **dimension** of the dataset. But this fact could give some unexpected behaviors and it is **strongly recommended** giving all parameters of offset and dimension.

The offset and dimension values must be equal to the number of rank. If rank=3 offset/dimension must contain 3 separated comma values (one for each rank) if offset/dimension contains only two values the third value is assumed to be equal to the size of the dataset. Example rank=3, dataset dimension (200,200,200). **--hyperslab datasetname 10,10 20,20** is considered as **--hyperslab datasetname 10,10,200 20,20,200**

The parameter file must have only one hyperslab in the hyperslab parameter. More rows with an hyperslab parameter must be given to describe more than one dataset.

Note in case of volume with more than one hyperslab: users are strongly suggested to give the same hyperslab extension (count).

In case of volume with specified hyperslab the --compx-y-z values are ignored. Datasets with different numbers of rows will produce columns with the number of rows equal to the maximum number of rows. The rows will be pads with **missingvalue** parameter.

Datasets with rank greather than 1000 cannot be read.

Datasets can represent tables or volumes. If a dataset represents a volume the dataset rank **must** be equal to 3 and the --volume option must be given.

## Tables

A dataset with rank > 1 will produce different columns in the VBT. If a dataset is a table and it has rank=3 and hyperslab offset=0,0,0 count=15,10,1000, it will produce 15\*10 columns each having 1000 elements. The columns names will be datasetname\_0\_0 datasetname\_0\_1 datasetname\_0\_2.... datasetname\_14\_9.

## Volume

The dataset rank must be equal to 3.

If the dataset represents a volume the --volume option must be given. In this case the hyperslab dimension (if given) represents the volume dimension and the --compx --compy --compz options are ignored.

More datasets can be given, but in this case they **must** have the same hyperslab dimension (the first hyperslab sets the volume resolution)

## Examples

### 1. Tables

Reading three datasets from a file., the dataset1 has rank = 2, the dataset2 has rank =3 and hyperslabs are specified. The dataset 3 is totally imported.

```
fformat=hdf5
datasetlist=dataset1
datasetlist=dataset2
datasetlist=dataset3
```

```
out=filename_out.bin
```

```
hyperslab=dataset1 20,45,65 32,38,49
hyperslab=dataset2 30,45,65 32,38,49
```

```
file=myFile.h5
```

### 2. Volume

Reading two datasets (volumes) from a file, each dataset must have rank = 3 and the same hyperslab extension.

```
fformat=hdf5
datasetlist=vol1
datasetlist=vol2
```

```
out=filename_out.bin
```

```
hyperslab=vol1 0,0,0 50,45,50
hyperslab=vol2 100,100,100 50,45,50
```

```
volume=true
compx=50
```

```
compy=45  
compz=50  
sizex=1.0  
sizey=1.0  
sizez=1.0  
  
file=myFile.h5
```



# VisIVO Importer

## MUPORTAL FORMAT

MUPORTAL files are expected to be in tabular form. They are produced from the experiment Muon Portal. It is an ASCII file containing rows with 10 values space separated. Each row represent an event (muon track). The column names are automatically added by the importer. The first row of the file contain the first event. The columns are typically separated by whitespace characters, e.g. spaces or tabs. The 10 columns represent: Event number, X\_A Y\_B X\_C Y\_D X\_E Y\_F X\_G Y\_H (8 values coordinates in cm at the planes of the system), Pulse energy in GeV/C. As an example, the command below produces *NewTable.bin*, *NewTable.bin.head* from *MuPortal.in*.

### Syntax Example

```
VisIVOImporter --fformat muportal --out /home/user/data/NewTable MuPortal.in
```

NOTE : The Importer automatically skips all lines starting with # character.

# VisIVO Importer

## RAMSES FORMAT

RAMSES files contain many type of data. This importer reads the particles positions only.

### Syntax Example

```
VisIVOImporter --fformat ramses --out /home/user/data/NewTable dir/output/part_seq
```

The input is the root filename of a sequence of files (normally equal to the number of processes that generate the ramses output). Each file contains a set of particles (e.g. part\_seq.out00001 .... part\_seq.out00064). The importer add *.outXXXXX* where XXXXX are 5 numbers in a sequence from 00001 to the number of processes that is automatically read from files.

The ramses file can have 1, 2 or 3 dimension. The generated output in any case will have 3 dimensions. In case of 1 or 2 dimensions the second and/or third dimension assume the missingvalue (--missingvalue option).

# VisIVO Importer

## RAW Binary FORMAT

Raw files are simply a binary dump of the memory for data points. The content of the Raw Binary data points file is a sequence of x,y and z coordinate for each point, then a sequence of fields, one scalar for each data point.

### General binary file structure:

X1, Y1, Z1  
X2, Y2, Z2  
.....  
Xn, Yn, Zn

Field0\_1  
Field0\_2  
.....  
Field0\_n

Field1\_1  
Field1\_2  
.....  
Field1\_n  
.....

Fieldm\_1  
Fieldm\_2  
.....  
Fieldm\_n

VisIVOImporter reads a descriptor file. More than one raw data file name can be described. The descriptor file has the following structure:

- rawPointDesc
- Variable name
- Variable type
- Time Variable (at the moment not used, but needed in the descriptor file)
- Number of particles
- The size of the box
- Endianism type (b=big endian or l=little endian)
- List of Ids of the data files (a number representing the order or the time) and names of the data files

### Example of Descriptor file

*rawPointsDesc*

```
dark
Float
time
130000
50 50 50
b
0.0 16ml_096
0.5 16ml_104
1.0 16ml_112
```

All the files listed in the descriptor file must be given. Each file will be converted and an internal binary table will be created for each listed file. Output files will have the same name of --out parameter +listedfilename+".bin" and ".bin.head". If a filename starts with *http://* or *ftp://* the remote file is downloaded. If the --userpwd option is given the username and password are used for remote access to the file. Downloaded files are temporarily copied in the same directory given in --out option and cleaned at the end of the import phase. The file *downloadedFilename\_VisIVO\_list* in the current directory contains information on the download operations.

### **Syntax Example**

```
VisIVOImporter --ffformat rawpoints rawpointsUserFile.desc
```

# VisIVO Importer

## RAW Grid FORMAT

Raw files are simply a binary dump of the memory. For volume data, only one quantity is expected to be stored in each file. The content of a volume file is a sequence of values: one value for each mesh point.

VisIVOImporter reads a descriptor file. More than one raw data file name can be described. The descriptor file has the following structure:

- rawGridDesc
- Variable Name
- Number of spatial dimensions
- Variable type
- Number of cells in the first dimension
- Number of cells in the second dimension
- Number of cells in the third dimension
- Time variable (in the present release not used, but required in the descriptor file)
- Endianism type (b=big endian or l=little endian)
- List of Ids of the data files (a number representing the order or the time) and names of the data files

### Example of Descriptor file:

```
rawGridsDesc
density
3
Float
64
64
64
time
l
0.0 JET8Xhj.f064.dat.pff
0.7 JET8Xeh.f064.dat.pff
2.1 JET8Tat.f064.dat.pff
```

All the files listed in the descriptor file must be given. Each file will be converted and an internal binary table will be created for each listed file. Output files will have the same name of --out parameter +listedfilename+".bin" and ".bin.head". If a filename start with *http://* or *ftp://* the remote file is downloaded. If the --userpwd option is given the username and password are used for remote access to the file.

Downloaded files are temporarily copied in the same directory given in --out option and cleaned at the end of the import phase. The file *downLoadedFilename\_VisIVO\_list* in the current directory contains information on the download operations.

**Syntax Example:**

**VisIVOImporter --ffformat rawgrids rawpointsUserFile.desc**

# VisIVO Importer

## XML DOC FORMAT

This importer cannot be used if VisIVO is compiled with LIGHT option.

VisIVOImporter can read an XML file as reported below and can download local or remote files and convert them into one or more output binary internal data format files. This document is a VOTable that must contain all the key information to download data. Each row in the TABLEDA <TR> ... </TR> will be a field of the output file. All the rows with the same species identifier are included in the same output file.

If the Arraysize field has only one value a data point field is assumed. If Arraysize has three values a volume field is assumed.

Downloaded files have the same name of the remote file + "\_VisIVOImporter\_" + *progressive number* . The produced files (binary internal data format) have the following filenames: --out file root + "\_" + *Species* + ".bin" and ".bin.head"

If a filename starts with *http://* or *ftp://* the remote file is downloaded. If the --userpwd option is given the username and password are used for remote access to the file.

Downloaded files are temporarily copied in the same directory given in --out option and cleaned at the end of the import phase. The file *downLoadedFilename\_VisIVO\_list* in the current directory contains information on the download operations.

### FIELD description

#### Format

The XML file could contain the description of each field of the local or remote file. If the XML file does not contain information on the fields it can refer to a specified file format. In this case the Format key can contain one of the following values:

*ascii*   *csv*   *fitstable*   *votable*   *gadget*   *hdf5*

If a different key value is given, the remote file will be considered a binary file and all the below field must be given.

#### Species

Name of species. Each row with the same Species Id will be a field of the same out table file.

**Field**

Column Name reported in the output table

**Unit**

Unit of the specified file.

**Offset**

Byte to be skipped for the first value of the field.

**Stride**

Byte to be skipped between two consecutive values of the same field

**Endianism**

LittleEndian or BigEndian

**Rank**

Rank of the field. The Importer produce one column for each rank

**Arraysizes**

Number of elements in the specified field. It must contain one value if the field is a data point, three values if the field is a volume. In the latter case each value represents the 3D mesh size.

**Type**

Internal data representation. Allowed values: float, double, long double, int, long int and long long int

**Precision**

Number of Bytes used for the above Type. It must match the same representation of the system where VisIVOImporter is running.

**Syntax Example:**

**VisIVOImporter --fformat xml userFile.xml**

*XML Example file*

```
<?xml version="1.0" encoding="UTF-8"?>  
<VOTABLE xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```



```

xmlns="http://www.ivoa.net/xml/VOTable/v1.1"
xsi:schemaLocation="http://www.ivoa.net/xml/VOTable/v1.1
http://www.ivoa.net/xml/VOTable/v1.1">
<DESCRIPTION>
    This is a first TVO xml (VOTable) prototype for Theoretical data
</DESCRIPTION>

<INFO ID="Ref" name="-ref" value="VOTx1"/>

<RESOURCE ID="ITVO_Catania_Server">

<DESCRIPTION>Theoretical data from Catania Cosmological Simulations</DESCRIPTION>

<LINK href="http://itvo.oact.inaf.it/"> </LINK>
<TABLE name="MyQueryResults" ID="Result">
    <PARAM name="Simulation0003_12" ID="mySimId" datatype="char" arraysize="*"
value="Simulation0003_12">
    </PARAM>
    <FIELD name="Time" UNIT="Redshift" datatype="float" ucd="" precision="4" utype=""/>
    <FIELD name="Format" datatype="char"/>
    <FIELD name="Species" datatype="char"/>
    <FIELD name="Field" datatype="char"/>
    <FIELD name="Unit" datatype="char"/>
    <FIELD name="Offset" datatype="long"/>
    <FIELD name="Stride" datatype="long"/>
    <FIELD name="Endianism" datatype="char"/>
    <FIELD name="Rank" datatype="int"/>
    <FIELD name="Arraysize" datatype="char"/>
    <FIELD name="Type" datatype="char"/>
    <FIELD name="Precision" datatype="int"/>
    <FIELD name="href" datatype="char"/>
    <DATA>
    <TABLEDATA>

<TR><TD>0.0</TD><TD>CUSTOM</TD><TD>DM</TD><TD>xcoordinate</TD><TD>Mpc
</TD><TD>0</TD><TD>8</TD><TD>LittleEndian</TD><TD>1</TD><TD>10000</TD>
<TD>float</TD><TD>4</TD><TD>ftp://astrct.oact.inaf.it/pub/becciani/Binary/out_2ml.bin
</TD></TR>

<TR><TD>0.0</TD><TD>CUSTOM</TD><TD>DM</TD><TD>ycoordinate</TD><TD>Mpc
</TD><TD>4</TD><TD>8</TD><TD>LittleEndian</TD><TD>1</TD><TD>10000</TD>
<TD>float</TD><TD>4</TD><TD>ftp://astrct.oact.inaf.it/pub/becciani/Binary/out_2ml.bin
</TD></TR>

<TR><TD>0.0</TD><TD>CUSTOM</TD><TD>DM</TD><TD>zcoordinate</TD><TD>Mpc
</TD><TD>8</TD><TD>8</TD><TD>LittleEndian</TD><TD>1</TD><TD>10000</TD>
<TD>float</TD><TD>4</TD><TD>ftp://astrct.oact.inaf.it/pub/becciani/Binary/out_2ml.bin
</TD></TR>

```

```
<TR><TD>0.0</TD><TD>CUSTOM</TD><TD>DM</TD><TD>mass</TD><TD>SolarMasses</TD><TD>120000</TD><TD>0</TD><TD>LittleEndian</TD><TD>1</TD><TD>10000</TD><TD>float</TD><TD>4</TD><TD>/home/user/ Binary/out_2ml.bin</TD></TR>
  </TABLEDATA>
</DATA>
</TABLE>
</RESOURCE>
</VOTABLE>
```

# VisIVO Importer

## VOTable FORMAT

The VOTable format is an XML standard for the interchange of data represented as a set of tables. In this context, a table is an unordered set of rows, each having a uniform format, as specified in the table metadata information. Each row in a table is a sequence of table cells, and each of these contain either a primitive data type or an array of such primitives. It can also contain a link to an external file, that the XML part describes. No VOTables with binary values are supported in VisIVO.

The file sizes that can be processed by the VisIVO Server are only limited by the underlying parsing libraries. As an example, the command below produces *NewTable.bin*, *NewTable.bin.head* from *VOTableUserFileName.xml*.

### Syntax Example:

```
VisIVOImporter --fformat votable --out /home/user/dataNewTable.bin  
VOTableUserFilename.xml
```

This reader has no limit on VOTable size, but can read only ascii data.